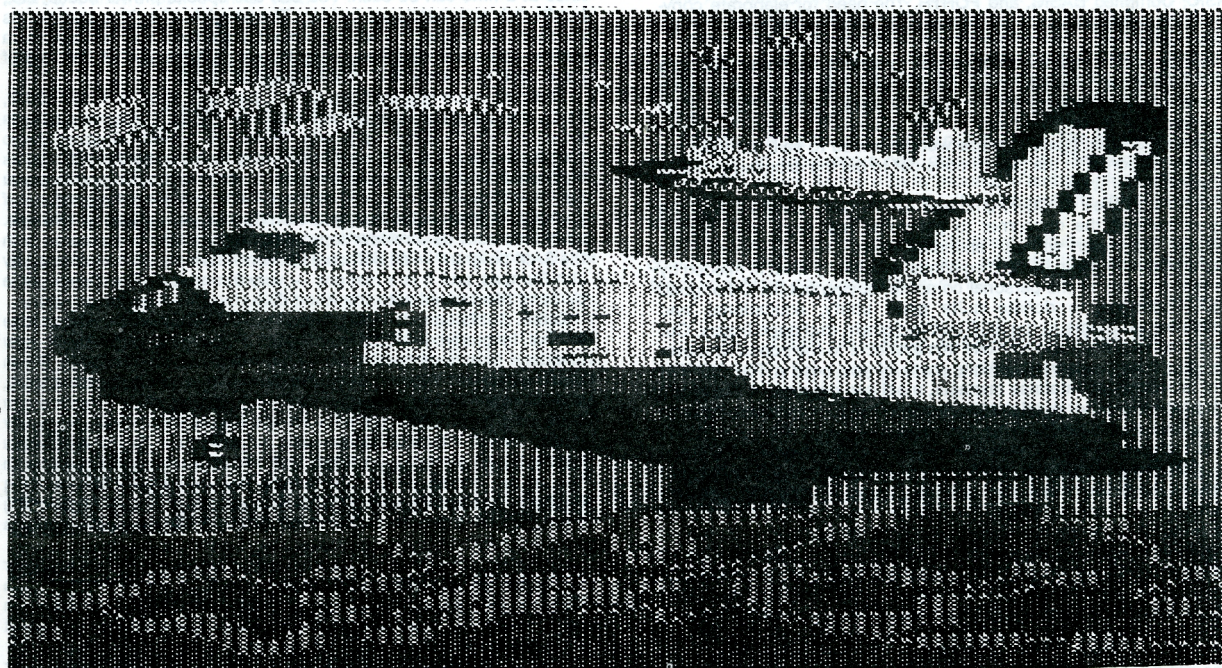


ACE ATARI COMPUTER ENTHUSIASTS

3662 Vine Maple Dr. Eugene OR 97405

MAY, 1983

Mike Dunn & Jim Bumpas, Editors



GTIA Sketchpad by Scott Berfield

Space Shuttle picture drawn by above program
dumped to an IDS MicroPrism printer

DIRECTIONS

Many items at the Computer Faire in San Francisco (reported in our last issue) intrigued me. The new equipment being produced at such low prices boggles my mind. The new games for the state-of-the-art personal computers are extremely well done. Of all the areas being advanced there is one which is not moving noticeably.

As fast as hardware and game software are moving, there is very little new being accomplished in the areas of APPLICATIONS and EDUCATIONAL software. Of course the programmers and publishers will produce and sell what the public is clamoring for. For a long time the clamor has not been for a better word processor but for a new game as good as the one in the arcades.

The tastes and needs of the American (and foreign) consuming public have been changing lately. With the budget cuts forcing schools to lower the flexibility (if not quality) of the education they provide, parents are becoming increasingly aware of the need to fill some gaps at home. The computer is ideally suited to this task, but the vast majority of programmers have yet to realize the potential financial rewards in this area. The same holds true for application software. With the cost of a capable microcomputer dropping regularly, and price projections showing enormous price reductions in the next year, it is inevitable the application software market will change drastically. Nobody will want to pay \$250.00 for Visicalc(tm) if that is all they paid for their 48K computer.

The cost of all types of programming is obviously going to have to be reduced by a large margin. Of course this should be offset by the larger numbers of people buying the programs. The quality and USER FRIENDLINESS of application-type programs will have to increase also. Consumers will not be eager to pay even reasonable prices for software which isn't as good as the computers have become. It is my belief the software publishing industry will change radically in the next few years. The emphasis on Machine Language programming will continue, but there will be more commercial programs available in BASIC, Compiled BASIC, and FORTH. The need for elaborate copy-protection schemes will decline when program prices reflect their true market value.

This is not to say the commercial programs currently available aren't quite good. I happen to use a commercial word-processor and a commercial database manager. I have, however, written many utility programs for myself, and am in the process of writing my own database manager. None of these were intended to sell, but were my personal response to the (what I consider to be) the excessive cost of commercial software.

Such magazines as COMPUTE and ANTIC have done much in the way of helping the consumer to get some real work from his/her personal computer without spending their grocery money. The programs published in those magazines are exactly what is needed by the vast majority of the personal computer-buying public. Other good people continue to donate quite good utility, educational, and application programs to the ACE public-domain exchange library. Our group makes every effort to make this available to the public wherever they might be in the world. While many of you need a TOP NOTCH word processor or database manager, most can get along quite nicely with programs either published in the magazines or available through the various computer clubs.

Let me make it clear I have no intention of insulting or denigrating the software publishers. Many of the commercially published products have been very high quality. There is undoubtedly some inertia needing to be overcome in shifting operations to accommodate the future. In fact, the software publishers have been largely responsible for the incredible success of the micro-computer movement. Without all of those captivating games, many owners would not now have their computers. If they didn't own their computers they would not be considering using a computer to make their work or education more efficient or rewarding.

I want to make a prediction. If FIRST RATE application software does not become available at much lower prices within the year the publishers of commercial software will lose a noticeable percentage of their market share. Many people will turn to enhancement devices, such as the ATR 8000 or the MICROMAINFRAME disk drive. These devices allow the purchase of inexpensive disk drives, and will allow the user to run extremely good software. In many cases, fairly complete software packages will come with the hardware, further lowering the cost to the consumer. It will be interesting to see what tactics the differing elements in this sales war take.

Enough for now; next month we will look at the future of HARDWARE and how it might affect you. If you feel a need to express your opinion about this article, my number is on the back page. If you want to write educational software and want solid guidelines, write to our ERACE sig. They will be happy to send you a copy of our SOFTWARE EVALUATION form if you send a SASE. This form can give you an excellent idea of what educators are looking for and what potential problems you should avoid in your programming. Good Luck to you all.

—Kirt E. Stockwell

News and Reviews

by Mike Dunn, Co-editor

Tara Keyboard

(2 Robert Speck Pky, Suite 1540, Mississauga, Ontario L42 1H8
or Statler Bld., 107 Delaware, Suite 1610, Buffalo, NY 14202)

Two months ago, I reviewed the InHome Keyboard for the Atari 400. After reading the review, Tara sent me theirs to compare. The Tara keyboard is also made in Canada, and is as easy to install as a direct replacement for the membrane one. Total time: less than 5 minutes. This keyboard is very well made, with a good feel and sturdy. The keys are laid out in a manner similar to the Inhome and different than the 400. It seems to stand high in relation to the computer, but actually measures somewhat lower than the 800. The return key is slightly closer to the L key, and can be reached easily. My only real criticism is the space bar is higher than the first row of keys, which takes some getting use to. I prefer this one, but some like the InHome better. You need to try both to find the one you like. Now our bulletin board 400 is completely Tara equipt — both the 48K memory board and the keyboard were donated by Tara — thanks!! The computer has been running 24 hours a day for several months without a glitch.

The Alog Pagewriter

(Alog computing, 1040 Veronica Springs Rd., Santa Barbara, CA 93105,
\$40)

This is a very unique new text processor written in valFORTH. The review copy is also unique — it allows you to edit, but when you try to print, you get "Buy alog pagewriter", so I really can't evaluate it. When the program is booted in, you get a picture of an 8" by 10" piece of paper, and a 2 line text window. As you type in the text window, dots are made on the "paper" corresponding to the letters and words, so you can see the page format at all times. For editing, etc., only two lines at a time can be seen. You can use special printer codes, etc., but Search commands are lacking. The usual Atari editing commands are there, as well as the ability to manipulate blocks of text and extra editing commands such as going to beginning or end of line, top of page, split line, etc.

The new voice box by Alien (27 W. 23rd ST., New York, NY 10010, \$150), the Alien II has just arrived and is very good. The voice quality is excellent, and it sings very well. There are several amazing demo programs which are very good, especially the ones by Jerry White, as well as some games and some fantastic songs with a singing face you can modify. Will be reviewed by the ERACE group next month, but if you want a good voice box, you won't go wrong by buying this one. Jerry also sent us a review copy of his new talking poker game with the SAM program — all software speech — which will be reviewed by Jim Bumpas, and it is a real gas — very funny. A similar poker game by Jerry is included with the Alien II.

O.S.S. (10379 Lansdale Ave, Cupertino, CA 95014) is about to release a new language ACTION on a 16K ROM. This remarkable new language is extremely fast; on bench marks such as in BYTE it can beat the fastest! You can do anything with it; somewhat similar to C, it comes with a few procedures and libraries of others, or you can write your own. Other projects on the fire at O.S.S. include a new BASIC, BASIC 400, which is sort of a stripped down BASIC A+ with a resident DOS on a plug in ROM. This allows an Atari 400 owner with 16K to run a disk drive! A new very fast Integer BASIC is being finalized, and an upgraded BASIC A+, possibly on a ROM, is being developed. This will be available to present BASIC A+ owners for a nominal price.

Adventure International has just announced a new Disassembler for \$50 allowing you to disassemble machine code which you can then load into any assembler and re-assemble, make changes, etc. It even puts the official Atari memory labels on it. I haven't seen it, but it sounds very good.

John Wiley & Sons (1 Wiley Drive, NJ 08873) has a new book in their computer programmed learning series. Written in the same style as Atari BASIC by Bob Albrecht, the book which used to come with the Atari, it covers 6502 assembly language very well. Designed for beginners, 6502 Assembly Language Programming by Judi Hernandez et al is an excellent introduction and guide to a very difficult (to some, like me!) subject. If you like to be forced to think, as programmed learning texts do, you will want this book. There are many self-tests, etc. This is a true self-teaching guide, rather than a textbook. Highly recommended.

Atari has finally released their new 850 Interface reference manual, and it is very good indeed. If you have an 850 interface and have not recieved yours, write to Atari at 1312 Crossman, Sunnyvale, CA 94086, and they will send it to you. AtariWriter is now in the stores at \$80 retail!! A best buy for sure. See my review in the March ACE or the May ANTIC! The printer drivers from APX have been delayed, but you can do anything except right justify proportional print without it.

The **ATR8000**, the disk drive/printer/RS232 interface with CP/M option, has been reduced to only \$350 for the basic 16K version. You get a 16K printer buffer, and, by adding an inexpensive generic, standard double-density drive, you have a great unit. You can upgrade to the full 64K CP/M version by yourself, or for \$600, get it all. They have also released their Co-Power-88, an add on allowing you to run IBM PC disks, and, when in the CP/M mode, lets you use the 8088 RAM for a RAM disk. Much of the software is now ready; however, the 850 interface emulation is not. They supply a modem program for CP/M use (Modem7), but you can not use the modem in the Atari mode yet. We have run KayPro and Osborne software with it.

We have some new disks this month for you. There are three education disks (see ERACE column), a disk of source codes which have been in ACE plus a PM utility and others, and an 1983 #1 disk. The Source code programs are assembly language listing, some in Atari and some in SynAssembler forms. Most are commented, and should prove useful to those of you interested in 6502 machine language. The 1983 #1 disk has all the programs from the newsletter since the christmas disk. The GTIA SketchPad has some nice screens done by the author (see front cover), and will be included. The Education Disks are \$10 each; the others are \$15 each. For double-sided disks, add \$5 for the extra side. **Any Atari drive can use double sided disks.**

All the money goes into the club — no one gets paid. Let me know if you want us to get another drive, so more programs can be put on the bulletin board, a 1200 BAUD modem, or whatever.

We have made the print on this issue larger, because of complaints about the small print. Let me know if this is large enough — but, of course, it allows less room for text.

—Mike Dunn, Co-Editor

Preppie II!

Review by Marc Russell Benioff
631 West 28th Street, Los Angeles, CA 90007

In 1982, Russ Whetmore designed Preppie! for Adventure International. Preppie was a humorous game involving a young prepster named, Wadsworth Overcash. At that time, Wadsworth was working to go to college at the local golf course. During his day, he had to avoid golf carts, alligators, and other assorted dangers. Finally, the summer was over and he enrolled at USC. Wadsworth knew all the true preps went to school there.

After Wadsworth entered USC, he learned about the fraternity rush, and looked at several houses. He found the best house on fraternity row, Tau Kappa Epsilon — The Tekes. He wanted to join TKE very badly, and for this reason they pledged Wadsworth. During his pledge period, Wadsworth was required to do many strange things, and finally his initiation week arrived. During this week the actives gave him his final goal. Wadsworth was to paint the top and bottom floors of the fraternity house, as well as the street adjacent to the house. Wadsworth was confident of his ability and so he began on this monstrous task.

In 1983, Russ Whetmore released Preppie II! In this new game, you are again Wadsworth. The object in this game is to paint the fraternity house, and the street. You must avoid frogs, cars, and other assorted menaces while completing your fraternity goal.

Preppie II is an excellent sequel. It uses all the aspects of the Atari required to become an excellent game. The music playing during the game is superb, and was created by the synthesizer for which Whetmore is famous. Even though the game may seem simplistic to some — this is one of the factors making Preppie II so good.

The colors used in Preppie II! follow the logic of the game — all of them preppie. Pinks, blues, and greens are used to maintain the full aspect of having a preppie game. The joystick response is superb, and the button allows you to start the game. The button can also be used to stop other tedious routines, such as the part of the game when you die and are carried away to heaven. This continuous user control just touches on the several parts of the human engineering which is built in.

Every time Russ releases a new game it turns to gold. He is currently working on Preppie III, and rumor has it it will be an anti-prep version. Russ Whetmore comes through on his third Atari game.

Preppie II!

Adventure International
Box 3435, Longwood, Florida 32750
\$34.95
ATARI 400/800/1200xl
16K Cass, 32K Disk

BUMPAS REVIEWS

POKERSAM is a 5-card stud poker game for the Atari by our friend Jerry White.

The game is produced by **Don't Ask Software** and requires no cartridge. The game is played by keyboard input alone. The title page plays a nice tune while you read. But the real treat is when you start the play.

I've always enjoyed poker. I never win because my hand can always be read all over my face. I figured I could stand a better chance with a mere computer. That might be true, except for the wise cracks 'Cincinnati Sam', or 'Poker Sam' (whatever he calls himself) makes all the time the program is running. It distracts you from the play when you try not to miss the next gem the computer is saying.

Oh, did I forget to tell you? The program includes voice synthesis routines (these are the same people who bring you 'S.A.M.', the software voice synthesizer). I played one session and counted nearly 50 wise-crack phrases uttered by Poker Sam. In addition, he talks up every play in the game. Whatever you do, or he does, he has to say it. The game is very entertaining if only for the computer generated voice and the content of the wise cracks.

By the way, Poker Sam plays a pretty good game of poker, too.

A.E. is a new arcade shoot 'em up from Broderbund Software. The game may be played with either joystick or paddle and requires 48k. The program is fully guaranteed and will be replaced for free unless the disk is physically destroyed (in which case they ask \$5).

You have a gun which moves across the bottom of the screen. The targets look like little Manta Rays chained together in a string. You shoot a kind of bomb which can destroy some or all of the AEs as they enter the barrage area.

The AEs may drop bombs to destroy your gun, or they may sweep down to the ground and eat you.

The box tells you the instructions are on the disk. I take this to mean I can read how to use all the game functions. I can't find the 'instructions', but if one just lets the program go without pushing the trigger it seems to cycle through the 5 screens, showing how the targets move.

The graphics are the real feature of this game. The music is nice but seems only to announce a new graphics page. The graphics have a high-res 3-D effect. As the AEs fly around, in and out of asteroid belts, planets and buildings you find the 'terrain' protects them from your shots.

The bombs you shoot go off at different altitudes. I think you can control the fusing of the bombs (i.e., with the joystick), but I don't seem to achieve any consistency so far.

—Jim Bumpas

A Look At the Commodore 64

For Christmas my father gave me a Commodore 64. I like it and my father says it has potential.

The 64 comes in a grey brown case. The case is the same as the VIC-20 case. The keys are black. But who cares very much how it looks? Let's look at the 64's features.

I asked a couple of my friends to tell me what they like best about the 64. Bill says he likes the graphics, and Will says, "I like the 64k memory at the low price."

The 64 has 64k memory, but the computer can't use all of it for BASIC. So, when you power up, BASIC says it has about 38k it can use. Also the RAM the operating system doesn't use is all yours for machine language programs.

The 64's graphics include a 300 by 200 high resolution graphics mode, a 160 by 200 medium resolution graphics mode, sprites, and some more text and graphics modes.

If you think Player Missiles are easy to use, just take a look at the 64's sprites. There is only one POKE for the X position, and only one POKE for the Y position. The sprites are considerably larger than Atari's. The 64 has a very big advantage over Atari with sprites.

The 64 sound is very hard to use. I wish you could program the sound on the 64 like you do on the Atari. Atari beats the 64 here.

As you know, the Atari has a lot of software going for it. Well, I haven't seen much software for the 64 (good or bad).

If you buy a 64, don't expect much support from Commodore. The User's Guide has only 166 pages, and it doesn't give enough information for a two year old. You had better count on spending \$20 for a reference manual which, by the way, is 486 pages long. It has enough information to last you for a couple of years.

—Jim Ockers

Old Mac Donald

Old Mac Donald is a program to aid preschoolers in learning the concepts of number. Animals appear in separate fenced off sections of a barnyard. When the animals on the right side match in some way the animals on the left, the child presses a key or the fire button on joystick 0. A correct match is rewarded with a smiley face while an incorrect response produces a frown. Smiley faces all the way across the top of the screen results in the playing of 'Old Mac Donald'.

The difficulty level can be set using the 'menu'. Animals appearing on the left and right can be the same (cows and cows) or different (cows and ducks). The positions at which animals appear can be ordered or they can be made to appear at random. Further, the maximum number of animals to appear in one set and the delay in waiting for a response can be adjusted.

There are three separate tasks to choose from. Perhaps the easiest is to match the total number of animals on the left and right (MATCH GROUPS). Another possibility is the COUNT ANIMALS task. In this case a certain number of animals appears on the left and animals appear on the right one at a time. The object is to wait until the number on the right equals the number on the left. Perhaps the most difficult task is to MATCH PATTERNS. In this case the number of animals left and right is always the same but they are scrambled in different positions. The object is to wait until the same animals appear in the same relative positions left and right (the patterns match).

My son (almost 3) seems to enjoy playing the game. He seems to get a little confused except on the simplest tasks and occasionally delights in filling the screen with 'uggy' faces, but I consider it a major accomplishment when he actually asks for the program. If you have any comments or suggestions for improvement I will appreciate hearing from you.

—Stan Ockers

RR#4 Box 209, Lockport, IL 60441

GTIA Sketchpad

by Scott Berfield

This program was sent in by Scott Berfield, a new member from Chicago. It is a fabulous painting program, with many features, but needs a GTIA chip to work. To make it even more fabulous, I've included subroutines which will dump your picture to an Epson or IDS MicroPrism printer; it should be easy to modify it to work with a NEC/Prowriter or other graphic printer. The subroutine is adapted from an article by Edward Schultz, Jr. reprinted from MACE, August, 1982, by way of Pat Warnshuis of the Portland Atari Club. This will print in half-tones, and the picture resembles a newspaper photograph. The picture on the front cover was done by Scott, and dumped to an IDS MicroPrism. If you don't want to type this all in, it's on our "1983 #1" disk or tape.

Commands:

Point use joystick to move cursor; press trigger to plot point.

Line Press trigger to plot one end, press trigger at other end to draw the line.

Doodle draw with joystick, press trigger for thick line; ESC for menu.

Box Plot one corner, move to opposite corner and press trigger to draw box.

Circle input the correction factor (to make up for the oblong Gr. 9-11 pixels). Plot the center. Move to a point on the circumference and press the trigger.

Fill area to be filled must be completely outlined in the fill color! Move cursor into the area and press the trigger.

Whole Screen wipes the entire drawing area with color.

Save input filename or C for cassette. E to cancel save.

Retrive input filename to be retrieved (Loaded).

Hardcopy dumps to IDS MicroPrism printer or Epson MX80 with Graphix with proper subroute.

New start over.

Quit quit.

Use the left and right arrow keys to change colors at any time- you must start with a color to see your drawing. ESC will return you to the menu from any command.

Since the program is in BASIC, you can change or add as you like. The author retains the rights to this one, but is allowing ACE members to use it.

—M. Dunn, Co-Editor

OCKERS

Machine Language # 6

Binary Operations

Binary math is similar to decimal math except there are only two digits (zero and one). Instead of the units, tens, hundreds, etc. places, we have the units, twos, fours, eights etc. places. Each place has a value double that of the one on the right rather than 10 times the value. The number 10101110 has the value (working right to left): $0 \times 1 + 1 \times 2 + 1 \times 4 + 1 \times 8 + 0 \times 16 + 1 \times 32 + 0 \times 64 + 1 \times 128 = 174$ decimal.

Numbers can be multiplied by two by shifting all digits to the left one place, bringing in a zero to the units place. The assembly language instruction to do this is ASL (Arithmetic Shift Left). The most frequently used ASL instruction works on the accumulator directly and has the op code 0A. Since the computer only works with 8 binary digits (BITS) at a time, the leftmost digit will be pushed out of the BYTE (8 bits). It is not lost but is pushed into a one bit location called the CARRY bit.

Division by two is quite simple also. All digits are shifted right one place. The units digit is lost and whatever occupies carry moves into the 128's place. The operation for this is called LSR (Logical Shift Right). The op code for LSR of the accumulator is 4A.

Two useful operations for combining numbers are AND and OR. To perform the operation, write the binary numbers on top of each other and inspect them by columns. In each column the result of an AND is a one only if both bits in that column are one. Any other combination gives a zero. The AND is useful for "masking" out part of a number. Simply AND the number with a MASK which has all bits you want to force to zero equal to zero. To zero out the lower four bits of 10101110 AND it with 11110000:

```
10101110 (initial byte)
11110000 (mask)
10100000 (result)
```

The OR operation gives a result of zero only if both bits in a column are zero. This makes it very useful in forcing certain bits to become ones. For example, OR the previous result with 00000111 (decimal 7):

```
10100000 (initial byte)
00000111 (mask)
10100111 (result)
```

All of these operations are used in a program I've written to provide sound/music independent of what is going on in Basic. The sounds themselves are stored in strings made up of consecutive frequency and duration bytes. A frequency byte of zero represents a rest while zero for both frequency and duration signals the end of the string.

A number of bytes in page 6 control the sounds and must be initialized properly. Most are in groups of 4, representing the four individual voices. Those labeled REPS control how many times a string will be repeated. Once the value reaches zero, the voice is no longer active. Poke 1666 with 3 and the string associated with voice 2 will be repeated 3 times. The location of strings is given to the routine by inserting the addresses in the bytes following STRLO and STRHI.

Counters determine how many times you pass through the VBI routine before another frequency-duration pair is loaded. Bytes holding counts start at CNT. The type of sound produced is determined by the distortion-volume bytes beginning at DIST. The lowest four bits of these bytes represent the volume. To separate notes in music these bits are modified after a certain count is reached, gradually reducing the volume to zero. The count at which this process begins is held in the byte QUIET. The higher the value of QUIET, the more separation between notes. If quiet is zero there will be no separation and sounds will run together. Non-music sounds require this. It is necessary to keep track of the volumes (SUSH and following) as well as the maximum volumes (LOUD and following) for the purpose of resetting the volumes.

A lot of the complication involved in the Sound in VBI routine occurs because of the order of frequency and control sound registers. Things would be a lot simpler if all frequency bytes were followed by all control bytes. Since they alternate it is necessary to double the index to reach every other byte.

Values for the positions in strings must be initialized once (STRPOS and following) but will be done automatically each time the sound is finished. The string is then ready for another 'playing'.

To use this routine it has to be set up in memory and inserted in the Vertical Blank Interrupt sequence. The program 'Old Mac Donald' uses the routine where it is held in the string VBI\$. Only four individual sound strings are used but more could be used by changing the pointers in page 6.

—Stan Ockers

RUTH'S PILOT

```

0 R:PILOT LISTING BY RUTH ELLSWORTH
1 R:ACE NEWSLETTER, 3662 VINE MAPLE
2 R: EUGENE, OR 97405 $10 YR
10 R:LOGO TYPE TURTLE
20 GR: CLEAR
30 *BEGIN
40 A:
50 M:U, D, R, L
60 JM:*UP,*DOWN,*RIGHT,*LEFT
70 E:
80 *TURTLE
90 GR: PENYELLOW
100 GR: TURN225;DRAW2;TURN180;GO 2;T
URN90;DRAW2;TURN180;GO 2;TURN-135;D
RAW1;TURN180;GO1
110 E:
120 *ERTURT
130 GR: PENERASE
140 GR: TURN225;DRAW2;TURN180;GO 2;T
URN90;DRAW2;TURN180;GO 2;TURN-135;T
URN180
150 GR: DRAW1
160 E:
170 *UP
180 U:*ERTURT
190 GR: PEN YELLOW
200 GR: TURNT00;DRAW1
210 U:*TURTLE
220 T: THIS IS TURNT00;DRAW 1
230 T:
240 J:*BEGIN
250 E:
260 *DOWN
270 U:*ERTURT
280 GR: PEN YELLOW
290 GR: TURNT0180;DRAW1
300 U:*TURTLE
310 T: THIS IS TURNT0180;DRAW 1
320 T:
330 J:*BEGIN
340 E:
350 *RIGHT
360 U:*ERTURT
370 GR: PEN YELLOW
380 GR: TURNT090;DRAW1
390 U:*TURTLE
400 T: THIS IS TURNT090;DRAW 1
410 T:
420 J:*BEGIN
430 E:
440 *LEFT
450 U:*ERTURT
460 GR: PEN YELLOW
470 GR: TURNT0-90;DRAW1
480 U:*TURTLE

```

```

490 T: THIS IS TURNT0-90 OR 270;DRAW
1
500 T:
510 J:*BEGIN
520 E:

```

```

10 R: THIS PROGRAM ALLOWS A YOUNG CH
ILD TO DRAW WITH THE TURTLE USING
ONLY THE DIRECTIONAL ARROWS

```

```

20 GR: CLEAR
30 GR: GOT00,0;TURNT00
40 *BEGIN
50 GR: PENYELLOW;4(TURN90;DRAW1)
60 A:=
70 J(@B764=14):*UP
80 J(@B764=15):*DOWN
90 J(@B764=6):*LEFT
100 J(@B764=7):*RIGHT
110 J(@B764=55):*FF
120 J(@B764=54):*MFF
130 J(@B764=34):*0TF
140 J(@B764=32):*TTF
150 GR: PENERASE;4(TURN90;DRAW1)
160 J:*BEGIN
170 E:
180 *UP
190 J(@Y>46):*SOUND
200 GR: TURNT00;DRAW 1
210 C:@B764=49
220 J:*BEGIN
230 E:
240 *DOWN
245 J(@Y<-30):*SOUND
250 GR: TURNT0180;DRAW 1
260 C:@B764=49
270 J:*BEGIN
280 E:
290 *LEFT
295 J(@X<-78):*SOUND
300 GR: TURNT0270;DRAW 1
310 C:@B764=49
320 J:*BEGIN
330 E:
340 *RIGHT
345 J(@X>78):*SOUND
350 GR: TURNT090;DRAW 1
360 C:@B764=49
370 J:*BEGIN
380 E:
450 *FF
455 J(@X>78):*SOUND
456 J(@Y>46):*SOUND
460 GR: TURNT045;DRAW 1

```

```

470 C:@B764=49
480 J:*BEGIN
490 E:
500 *MFF
505 J(@X<-78):*SOUND
506 J(@Y>46):*SOUND
510 GR: TURNT0-45;DRAW 1
520 C:@B764=49
530 J:*BEGIN
540 E:
550 *0TF
555 J(@X>78):*SOUND
556 J(@Y<-30):*SOUND
560 GR: TURNT0135;DRAW 1
570 C:@B764=49
580 J:*BEGIN
590 E:
600 *TTF
605 J(@X<-78):*SOUND
606 J(@Y<-30):*SOUND
610 GR: TURNT0225;DRAW 1
620 C:@B764=49
630 J:*BEGIN
640 E:
650 *TOOUP
660 C:@Y=47
670 GR: DRAWTO#X,#Y
680 SO:13
690 PA:10
700 C:@B764=49
710 SO:0
720 J:*BEGIN
730 E:
800 *TOODOWN
810 C:@Y=-31
820 GR: DRAWTO#X,#Y
830 SO:13
840 PA:10
850 C:@B764=49
860 SO:0
870 J:*BEGIN
880 E:
900 *TOORIGHT
910 C:@X=79
920 GR: DRAWTO#X,#Y
930 SO:13
940 PA:10
950 C:@B764=49
960 SO:0
970 J:*BEGIN
980 E:
1000 *TOOLEFT
1010 C:@X=-79
1020 GR: DRAWTO#X,#Y

```

GOTO page 11

Berfield's Sketchpad

**GTIA SKETCHPAD
by Scott Berfield**

```

1 REM *****
2 REM *      ACE NEWSLETTER      **
3 REM *      3662 VINE MAPLE DR  **
4 REM *      EUGENE, OR 97405    **
5 REM *      $10 YR              **
6 REM *****
7 REM *      GTIA Sketchpad      **
8 REM *      by                  **
9 REM *      Scott Berfield      **
10 REM *      644 W. Surf, #405   **
11 REM *      Chicago, Ill 60657  **
12 REM *      (c) by the Author   **
13 REM *      all rights reserved **
14 REM *****
20 GRAPHICS 0:POKE 82,1:POKE 752,1
30 GOSUB 1520:GOSUB 1260
40 ? CHR$(125);CHOICE$;CHOICE$:LOC
ATE XP,YP,VAL:COLOR C:PLOT XP,YP
50 OPT=PEEK(KEY):IF OPT<>NOKEY THEN
POKE KEY,NOKEY:GOSUB 70
60 COLOR ABS(C-8):PLOT XP,YP:COLOR
C:PLOT XP,YP:COLOR VAL:PLOT XP,YP:G
OTO 50
70 IF OPT=10 THEN GOTO 580
80 IF OPT=NO THEN GOTO 620
90 IF OPT=58 THEN GOTO 660
100 IF OPT=21 THEN GOTO 840
110 IF OPT=18 THEN GOTO 880
120 IF OPT=56 THEN GOTO 210
130 IF OPT=62 THEN GOTO 1070
140 IF OPT=40 THEN GOTO 1130
150 IF OPT=47 THEN GOTO 1200
160 IF OPT=7 THEN GOTO 490
170 IF OPT=6 THEN GOTO 520
180 IF OPT=35 THEN GOTO 1200
185 IF OPT=57 THEN GOTO 32700:REM F
OR PRINTER DUMPING ONLY
190 IF OPT=46 THEN 1250
200 GOTO 50
210 ? CHR$(125);"PLACE CURSOR WITHI
N THE OUTLINE AND PRESS TRIGGER.
"
220 GOSUB JOYSTICK
230 COLOR C
240 YP1=YP:LEFT=270:DOWN=320:FLAG=0
:DFLAG=0
250 FOR Q=XP TO 319-XP:LOCATE Q,YP,
COLR:IF COLR=C THEN GOTO LEFT
260 PLOT Q,YP:NEXT Q
270 FOR Q=XP-1 TO 1 STEP -1:LOCATE.
Q,YP,COLR:IF COLR=C THEN 290
280 PLOT Q,YP:NEXT Q
290 IF FLAG=1 THEN GOTO DOWN
300 YP=YP-1:LOCATE XP,YP,COLR:IF CO
LR=C THEN FLAG=1:GOTO DOWN
310 GOTO 250
320 IF DFLAG=1 THEN 340
330 YP=YP1:DFLAG=1
340 YP=YP+1:LOCATE XP,YP,COLR:IF CO
LR=C THEN GOTO 40
350 GOTO 250
360 GOTO 40
370 IF STRIG(NO)=NO THEN COLOR C:RE
TURN
380 IF PEEK(KEY)=6 THEN POKE KEY,NO
KEY:GOSUB 520
390 IF PEEK(KEY)=7 THEN POKE KEY,NO
KEY:GOSUB 490
400 IF PEEK(KEY)=28 THEN POKE KEY,N
OKEY:GOTO 40
410 OXP=XP:OYP=YP:OVAL=VAL
420 XP=XP+XD(STICK(NO)):YP=YP+YD(ST
ICK(NO))
430 IF XP>79 THEN XP=0
440 IF XP<0 THEN XP=79
450 IF YP>149 THEN YP=0
460 IF YP<0 THEN YP=149
470 LOCATE XP,YP,VAL:COLOR OVAL:PLO
T OXP,OYP
480 COLOR C:PLOT XP,YP:COLOR ABS(C-
8):PLOT XP,YP:COLOR VAL:PLOT XP,YP:
GOTO 370
490 OB=B:OB1=OB+N1:OC=C:B=B+N2:C=C+
N1:IF C>CMAX THEN C=CMIN
500 IF B>BMAX THEN B=BMIN
510 GOTO 540
520 OB=B:OB1=OB+N1:OC=C:B=B-N2:C=C-
N1:IF C<CMIN THEN C=CMAX
530 IF B<BMIN THEN B=BMAX
540 COLOR OC:PLOT BAR(OB),159:DRAWT
O BAR(OB1),159
550 IF C<INT(0.5*CMAX)+N1 THEN COLO
R CMAX
560 IF C>INT(0.5*CMAX)+N1 THEN COL
OR CMIN
570 PLOT BAR(B),159:DRAWT O BAR(B+N1
),159:COLOR C:RETURN
580 ? CHR$(125);"POSITION CURSOR AN
D PRESS TRIGGER TO PLOT POINT."
590 GOSUB JOYSTICK
600 PLOT XP,YP
610 GOTO 40
620 ? CHR$(125);"POSITION CURSOR AN
D PRESS TRIGGER TO PLOT STARTING
POINT."
630 GOSUB JOYSTICK:PLOT XP,YP:CXP=X
P:CYP=YP
640 ? CHR$(125);"NOW POSITION CURSO
R AT ENDPOINT AND PRESS TRIGGER
TO DRAW LINE."
650 GOSUB JOYSTICK:PLOT CXP,CYP:DRA
WTO XP,YP:GOTO 40
660 ? CHR$(125);"USE JOYSTICK TO DR
AW, HOLD TRIGGER FOR A THICK LINE,
PRESS ESC TO EXIT TO MENU"
670 Z=PEEK(KEY):IF Z<>NOKEY THEN PO
KE KEY,NOKEY
680 IF Z=6 THEN GOSUB 520
690 IF Z=7 THEN GOSUB 490
700 IF Z=28 THEN GOTO 40
710 IF STRIG(NO)=NO THEN 770
720 XP=XP+XD(STICK(NO)):YP=YP+YD(ST
ICK(NO)):IF XP<NO THEN XP=79
730 IF XP>XMAX THEN XP=NO
740 IF YP<NO THEN YP=YMAX
750 IF YP>YMAX THEN YP=NO
760 COLOR ABS(C-CMAX):PLOT XP,YP:CO
LOR C:PLOT XP,YP:GOTO 670
770 IF STRIG(NO) THEN 670
780 XP=XP+XD(STICK(NO)):YP=YP+YD(ST
ICK(NO)):IF XP>78 THEN XP=N1
790 IF XP<N1 THEN XP=N1
800 IF YP<N1 THEN YP=78
810 IF YP>148 THEN YP=N1
820 PLOT XP-N1,YP-N1:DRAWT O XP+N1,Y
P-N1:PLOT XP+N1,YP:DRAWT O XP-N1,YP:
PLOT XP-N1,YP+N1:DRAWT O XP+N1,YP+N1
830 GOTO 770
840 ? CHR$(125);"POSITION CURSOR TO
LOWER RIGHT CORNER OF BOX AND PRE
SS TRIGGER."
850 GOSUB JOYSTICK:PLOT XP,YP:BX=XP
:BY=YP:? CHR$(125);"NOW POSITION TO
UPPER LEFT AND PRESS TRIGGER TO
PLOT";
870 GOSUB JOYSTICK:PLOT XP,YP:DRAWT
O BX,YP:DRAWT O BX,BY:DRAWT O XP,BY:D
RAWT O XP,YP:GOTO 40
880 TRAP 880:? CHR$(125);"ENTER ROU
NDNESS(1=3 TO 1 ELLIPSE, .275=ROUND
)";:INPUT SCALE
890 ? "NOW PLACE THE CURSOR AT THE
CENTER OF THE CIRCLE AND PRESS THE
TRIGGER.":GOSUB JOYSTICK:XC=XP:YC=
YP
900 ? CHR$(125);"NOW POSITION THE C
URSOR TO A POINT ON THE CIRCUMFERA
NCE OF THE CIRCLE AND PRESS THE
TRIGGER."

```



```

910 GOSUB JOYSTICK:XR=XP:YR=YP:IF X
C>XR THEN XD=XC-XR
920 IF XC>XR THEN XD=XC-XR:GOTO 940
930 XD=XR-XC
940 IF YC>YR THEN YD=YC-YR:GOTO 960
950 YD=YR-YC
960 IF XC=XR THEN R=YD:GOTO 990
970 IF YC=YR THEN R=XD:GOTO 990
980 R=SQR((XD*XD)+(YD*YD))
990 X=0:Y=R:DIAMETER=3-2*R
1000 IF X<Y THEN GOSUB 1040:IF DIA
METER<0 THEN DIAMETER=DIAMETER+4*X+
6:X=X+1:GOTO 1000
1010 IF X>Y THEN TRAP 40000:GOTO 40
1020 DIAMETER=DIAMETER+4*(X-Y)+10
1030 Y=Y-1:X=X+1:GOTO 1000
1040 TRAP 1060:PLOT XC+(X*SCALE),YC
+Y:PLOT XC+(Y*SCALE),YC+X:PLOT XC+(
Y*SCALE),YC-X:PLOT XC+(X*SCALE),YC-
Y
1050 PLOT XC-(X*SCALE),YC-Y:PLOT XC
-(Y*SCALE),YC-X:PLOT XC-(Y*SCALE),Y
C+X:PLOT XC-(X*SCALE),YC+Y
1060 RETURN
1070 TRAP 1070:? CHR$(125);"INPUT F
ILE NAME (E TO EXIT).":INPUT FILE$:
IF FILE$(1,1)="E" THEN TRAP 40000:G
OTO 40
1080 OPEN #N1,8,N0,FILE$:FOR I=704
TO 712:PUT #N1,PEEK(I):NEXT I
1090 RTOP=256*PEEK(106):BASE=PEEK(8
8)+256*PEEK(89):BT5=RTOP-BASE:HI=IN
T(BT5/256):LO=BT5-(HI*256)
1100 POKE 850,11:POKE 852,PEEK(88):
POKE 853,PEEK(89):POKE 856,LO:POKE
857,HI
1110 DUM=USR(ADR("hhh*LVd"),16):CLO
SE #N1:REM * AND d inverse
1120 TRAP 40000:GOTO 40
1130 TRAP 1130:? CHR$(125);"LOAD FR
OM WHICH FILE (E FOR MENU)":INPUT F
ILE$
1140 IF FILE$(1,1)="E" THEN TRAP 40
000:GOTO 40
1150 OPEN #N1,4,N0,FILE$
1160 FOR I=704 TO 712:GET #N1,A:POK
E I,A:NEXT I
1170 POKE 850,7:POKE 852,PEEK(88):P
OKE 853,PEEK(89):POKE 856,NOKEY:POK
E 857,NOKEY
1180 DUM=USR(ADR("hhh*LVd"),16):CLO
SE #N1:REM * AND d inverse
1190 TRAP 40000:GOTO 40

```

```

1200 ? CHR$(125);"DO YOU WANT TO SA
VE THIS IMAGE FIRST?":OPEN #N2,4,N0
,"K":GET #N2,A
1210 IF CHR$(A)<>"Y" AND CHR$(A)<>"
N" THEN CLOSE #N2:GOTO 1200
1220 IF CHR$(A)="Y" THEN GOSUB 1070
1230 IF OPT=47 THEN END
1240 IF OPT=35 THEN RUN
1250 COLOR C:FOR X2=N0 TO 79:PLOT X
2,N0:DRAWTO X2,YMAX:NEXT X2:GOTO 40
1260 REM *** MODE SELECTION ***
1270 ? CHR$(125)
1280 ? :? "PLEASE SELECT A GRAPHICS
MODE:"
1290 ? :? "1 MODE 9#16 SHADES OF O
NE HUE"
1300 ? :? "2 MODE 10#9 HUE/SHADE C
OMBINATIONS"
1310 ? :? "3 MODE 11#16 HUES OF ON
E SHADE":?
1320 OPEN #N2,4,N0,"K":GET #N2,A
1330 CLOSE #N2:ON A-48 GOTO 1340,13
70,1460
1340 MODE=9:? CHR$(125);"GRAPHICS M
ODE 9 GIVES YOU 16 SHADES OF ANY ON
E HUE."
1350 ? :? "WHAT BACKGROUND COLOR WO
ULD YOU LIKE TO USE(0-15)":INPUT
BKGND:GRAPHICS 8:POKE 752,1
1360 POKE 623,64:POKE 87,MODE:BMIN=
N1:BMAX=31:CMAX=15:SETCOLOR 4,BKGND
,N0:GOSUB 1600:B=N1:GOTO SETUP
1370 MODE=10:? CHR$(125);"GRAPHICS
MODE 10 GIVES YOU YOUR CHOICE OF AN
Y 9 HUE/SHADE COMBINATIONS.":?
1380 FOR Q=N0 TO 8
1390 POSITION N2,15:? "COLOR #";Q;"
HUE,SHADE(0-15) *****":INPU
T H,L
1400 IF H<N0 OR H>15 OR L<N0 OR L>1
5 THEN 1390
1410 C(Q)=H*16+L:NEXT Q
1420 BMIN=9:BMAX=25:CMAX=8:GRAPHICS
8:POKE 623,128:POKE 87,MODE:FOR Q=
704 TO 712:POKE Q,C(Q-704):NEXT Q
1430 POKE 752,1:GOSUB 1600
1450 B=BMIN:GOTO SETUP
1460 MODE=11:? CHR$(125);"GRAPHICS
MODE 11 GIVES YOU 16 HUES OF ANY ON
E SHADE.":?
1470 ? :? "WHAT SHADE(0-15) WOULD Y
OU LIKE":INPUT SHD

```

```

1480 GRAPHICS 8:POKE 752,1:POKE 623
,192:POKE 87,MODE:GOSUB 1600:SETCOL
OR 4,N0,SHD:BMIN=N1:BMAX=31:CMAX=15
:B=N1
1490 C=CMIN:COLOR C:FOR I=BAR(BMIN)
TO BAR(BMAX) STEP 4:FOR J=0 TO 3:P
LOT I+J,150:DRAWTO I+J,159:NEXT J
1500 C=C+N1:COLOR C:NEXT I:C=CMIN
1510 COLOR C:XP=39:YP=79:COLOR ABS(
C-CMAX):PLOT BAR(BMIN),159:DRAWTO B
AR(BMIN+N1),159:RETURN
1520 POKE 559,0
1530 DIM XD(15),YD(15),BAR(32),C(8)
,CHOICE$(80),CHOICE1$(40),FILE$(20)
1540 CMIN=N0:N0=0:N1=1:N2=2:XMAX=79
:YMAX=149:SCALE=0.275:SETUP=1490:JO
YSTICK=370:KEY=764:NOKEY=255
1550 FOR Q=1 TO 15:READ N:X D(Q)=N:R
EAD N:Y D(Q)=N:NEXT Q
1560 DATA 0,0,0,0,0,0,0,0,1,1,1,-1,
1,0,0,0,-1,1,-1,-1,-1,0,0,0,1,0,-
1,0,0
1570 FOR Q=1 TO 32:READ N:BAR(Q)=N:
NEXT Q
1580 DATA 8,11,12,15,16,19,20,23,24
,27,28,31,32,35,36,39,40,43,44,47,4
8,51,52,55,56,59,60,63,64,67,68,71
1590 CHOICE$="Point, Line, Doodle,
Box, Circle, Fill, Whole screen, Sav
e, Retrieve"
1591 CHOICE1$=" Hardcopy,
New, Quit":POKE 559,34:RETURN
:REM Use INVERSE to begin each word
1600 DL=PEEK(560)+256*PEEK(561)
1610 POKE 559,0
1620 POKE DL+166,143
1630 POKE 513,6:POKE 512,0
1640 FOR I=1536 TO 1546
1650 READ A:POKE I,A:NEXT I
1660 DATA 72,169,0,141,10,212,141,2
7,208,104,64
1670 POKE 54286,192
1680 POKE 559,34:RETURN
32690 REM PRINTER SUBROUTINE ADAPTED
FROM RON MILLER, SAN DIEGO ACE, FOR
IDS Microprism PRINTER
32700 DIM GREY$(32),BUFFER$(400):CL
OSE #7:OPEN #7,8,0,"P":PRINT #7,CH
R$(3):PRINT #7
32701 FOR I=1 TO 32:READ A:GREY$(I,
I)=CHR$(A):NEXT I:REM Set grey scal
e
32702 DATA 255,255,251,223,251,222,
206,183

```


OCKERS!

OLD MACDONALD
by Stan Ockers

```
0 REM *****
1 REM * ACE NEWSLETTER *
2 REM * 3662 VINE MAPLE DR *
3 REM * EUGENE, OR 97405 *
4 REM * $10 YR *
5 REM *****
6 REM *****
7 REM * OLD MACDONALD, a game *
8 REM * for children *
9 REM * by *
10 REM * Stan Ockers *
11 REM * Lockport, IL *
12 REM ** May, 1983 **
13 REM *****
100 GOSUB 2080:GOSUB 650
110 OPEN #1,4,0,"K":DLY=3:LIM=5:GO
SUB 1830
120 GRAPHICS 0:?"INITIALIZING":?"
TAKES 15 SECONDS"
130 DIM X1(8),Y1(8),X2(8),Y2(8),A1(
8),A2(8),A$(5),F1$(40),F2$(40),B$(1
7):B$(1)="":B$(17)="":B$(2)=B$
140 RESTORE 150:FOR J=0 TO 8:READ A
,B,C,D:X1(J)=A:Y1(J)=B:X2(J)=C:Y2(J
)=D:NEXT J
150 DATA 3,6,22,6,8,6,27,6,13,6,32,
6,5,9,24,9,12,9,31,9,4,12,23,12,10,
12,29,12,6,15,25,15,12,15,31,15
160 F1$(1)="kmm":F1$(39)="M":F1$(4)
=F1$:F2$(1)="lmm":F2$(39)="M":F2$(4)
=F2$
170 DIM F3$(17),B1(8),B2(8):F3$="xx
xxjopjjopjjopjj"
180 GOSUB 1010:GOSUB 1290:POKE 16,1
12:POKE 53774,112
190 SMILY=0:FROWN=0:?"CHR$(125):SOU
ND 2,0,0,0:SOUND 3,0,0,0
200 POSITION 0,3:?"F1$":POSITION 0,
4:?"F2$":POSITION 0,18:?"F1$":POSIT
ION 0,19:?"F2$";
210 FOR J=5 TO 17:POSITION 1,J:?"F3
$(J,J)":POSITION 20,J:?"F3$(J,J)":P
OSITION 38,J:?"F3$(J,J)":NEXT J
220 ANIMAL=INT(RND(0)*7)+1:NUM=INT(
RND(0)*(LIM-1))+2:ANIMAL2=ANIMAL+1:
IF ANIMAL2=8 THEN ANIMAL2=1
230 IF LOCN=0 THEN FOR J=0 TO 8:A1(
J)=J:A2(J)=J:NEXT J
240 IF LOCN=1 THEN GOSUB 1350
250 IF TYPE=0 THEN ANIMAL2=ANIMAL
260 GOSUB 1570:GOSUB 1580
```

270 POSITION 2,20:?"#6:"

```
"
280 IF TASK(<>) THEN 390
290 REM * MATCH GROUPS *
300 GOSUB 1700
310 GOSUB 610:FOR J=0 TO R-1:GOSUB
1430:NEXT J
320 GRP=-1
330 GOSUB 1580:GRP=GRP+1:IF GRP>LIM
-2 THEN GOSUB 1650:GOTO 220
340 FOR J=0 TO B1(GRP)-1:GOSUB 1400
:NEXT J:N=B1(GRP):GOSUB 1550
350 MAXTIME=60*DLY:GOSUB 1520:GOSUB
1590
360 IF TIMEUP=1 THEN 330
370 IF B1(GRP)<>R THEN GOSUB 1650:G
OTO 220
380 GOSUB 1770:GOTO 220
390 IF TASK(<>) THEN 490
400 REM * COUNT ANIMALS *
410 FOR J=0 TO NUM-1:GOSUB 1430:NEX
T J
420 CNT=0:MAXTIME=60*DLY:GOSUB 1520
430 GOSUB 1590:IF TIMEUP=0 THEN 460
440 J=CNT:GOSUB 1400:GOSUB 1530:CNT
=CNT+1:N=CNT:GOSUB 1550:GOSUB 1520
450 IF CNT<NUM OR CNT=NUM THEN 430
460 IF CNT=NUM THEN GOSUB 1770:GOTO
220
470 GOSUB 1650:GOTO 220
480 REM * MATCH PATTERNS *
490 NUM=LIM:GOSUB 1350:GOSUB 1730:R
AND=INT(RND(0)*(LIM-1))+2:ANIMAL=RA
ND
500 FOR J=0 TO LIM-1:ANIMAL=ANIMAL+
1:IF ANIMAL>7 THEN ANIMAL=1
510 GOSUB 1430:NEXT J
520 SEQN=-1:FOR J=0 TO 8:A2(J)=A1(J
):NEXT J
530 GOSUB 1580:SEQN=SEQN+1:IF SEQN>
LIM THEN GOSUB 1650:GOTO 220
540 ANIMAL2=B1(SEQN):FOR J=0 TO LIM
-1:ANIMAL2=ANIMAL2+1:IF ANIMAL2>7 T
HEN ANIMAL2=1
550 GOSUB 1400:NEXT J
560 MAXTIME=60*DLY:GOSUB 1520:GOSUB
1590
570 IF TIMEUP=1 THEN 530
580 IF B1(SEQN)<>RAND THEN GOSUB 16
50:GOTO 220
590 GOSUB 1770:GOTO 220
600 REM * RANDOM #, 2 TO LIM *
610 R=INT(RND(0)*(LIM-1))+2:RETURN
620 REM * RANDOM #, 1 TO LIM *
```

```
630 R=INT(RND(0)*LIM+1):RETURN
640 REM * VBI ROUTINE *
650 DIM VBI$(128):RESTORE 660:FOR J
=1 TO 128:READ A:VBI$(J,J)=CHR$(A):
NEXT J
660 DATA 162,3,189,128,6,240,115,22
2,132,6,48,39,189,132,6,205,160,6,1
76,102,189,152,6,240,97,189,148,6
670 DATA 41,240,29,152,6,72,134,205
,138,10,170,104,157,1,210,166,205,2
22,152,6,24,144,71,189,136,6,133,20
3
680 DATA 189,140,6,133,204,189,148,
6,133,206,188,144,6,134,205,138,10,
170,177,203,240,32,157,0,210,165,20
6
690 DATA 157,1,210,166,205,200,177,
203,240,23,157,132,6,200,152,157,14
4,6,189,156,6,157,152,6,24,144,12
700 DATA 157,0,210,24,144,223,157,1
44,6,222,128,6,202,16,133,76,98,228
710 REM * INSERT VBI ROUTINE *
720 RESTORE 740:FOR J=1536 TO 1545:
READ A:POKE J,A:NEXT J
730 VBI=ADR(VBI$):HI=INT(VBI/256):L
O=VBI-256*HI:POKE 1538,LO:POKE 1540
,HI
740 DATA 104,160,0,162,0,169,7,76,9
2,228
750 REM * SOUND STRINGS *
760 DIM SND1$(110):RESTORE 770:FOR
J=1 TO 110:READ A:SND1$(J,J)=CHR$(A
):NEXT J
770 DATA 81,20,81,20,81,20,108,20,9
6,20,96,20,108,40,64,20,64,20,72,20
,72,20,81,60,108,20,81,20,81,20,81,
20
780 DATA 108,20,96,20,96,20,108,40,
64,20,64,20,72,20,72,20,81,60,108,2
0,81,20,81,20,81,20,108,20,81,20,81
,20
790 DATA 81,20,108,20,81,20,108,20,
81,20,108,20,81,20,81,20,96,20,108,
20,81,20,81,20,81,20,108,20,96,20,9
6,20
800 DATA 108,40,64,20,64,20,72,20,7
2,20,81,60,0,0
810 DIM SND2$(98):RESTORE 820:FOR J
=1 TO 98:READ A:SND2$(J,J)=CHR$(A):
NEXT J
820 DATA 128,20,128,20,128,40,121,2
0,121,20,128,40,108,20,108,20,121,2
0,121,20,128,60,0,20
```



```

830 DATA 128,20,128,20,128,40,121,2
0,121,20,128,40,108,20,108,20,121,2
0,121,20,128,60,0,20,128,20,128,20,
128,20
840 DATA 0,20,128,20,128,20,128,20,
0,20,128,40,128,40,128,20,128,20,12
1,40,128,20,128,20,128,40,121,20,12
1,20
850 DATA 128,40,108,20,108,20,121,2
0,121,20,128,60,0,0
860 DIM SND3$(16):RESTORE 870:FOR J
=1 TO 16:READ A:SND3$(J,J)=CHR$(A):
NEXT J
870 DATA 96,20,81,20,72,20,81,20,96
,20,81,20,60,60,0,0
880 DIM SND4$(16):RESTORE 890:FOR J
=1 TO 16:READ A:SND4$(J,J)=CHR$(A):
NEXT J
890 DATA 102,20,108,20,102,20,121,2
0,162,20,128,20,121,60,0,0
900 REM * PAGE 6 INIT. *
910 RESTORE 920:FOR J=1664 TO 1696:
READ A:POKE J,A:NEXT J
920 DATA 1,1,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,170,170,170,170,10,10
,10,10,10,10,10,0
930 POKE 1696,15
940 HI=INT(ADR(SND1$)/256):POKE 167
6,HI:LO=ADR(SND1$)-256*HI:POKE 1672
,LO
950 HI=INT(ADR(SND2$)/256):POKE 167
7,HI:LO=ADR(SND2$)-256*HI:POKE 1673
,LO
960 HI=INT(ADR(SND3$)/256):POKE 167
8,HI:LO=ADR(SND3$)-256*HI:POKE 1674
,LO
970 HI=INT(ADR(SND4$)/256):POKE 167
9,HI:LO=ADR(SND4$)-256*HI:POKE 1675
,LO
980 A=USR(1536)
990 RETURN
1000 REM * Change character set *
1010 DIM MCS$(42):RESTORE 1020:FOR
J=1 TO 42:READ A:MCS$(J,J)=CHR$(A):
NEXT J
1020 DATA 104,169,0,133,203,133,205
,169,224,133,204,165,106,56,233,5,1
33,106,24
1030 DATA 105,1,133,206,162,4,160,0
,177,203,145,205,200,208,249,230,20
4,230,206,202,208,242,96
1040 A=USR(ADR(MCS$)):CSPAGE=PEEK(1
06)+1:C5=256*CSPAGE:GRAPHICS 0:POKE
756,CSPAGE

```

```

1050 RESTORE 1080:FOR J=C5+208 TO C
5+511:READ A:POKE J,A:NEXT J
1060 RESTORE 1210:FOR J=C5+776 TO C
5+927:READ A:POKE J,A:NEXT J
1070 RETURN
1080 DATA 0,165,41,10,2,0,0,0,0,85,
85,85,149,170,195,243,2,66,106,104,
160,128,0,192
1090 DATA 160,47,160,0,0,0,0,0,21
,165,165,133,134,134,134,0,85,85,85
,169,170,166,170
1100 DATA 0,90,90,90,86,86,86,85,25
3,21,145,149,147,159,175,169,127,84
,68,84,196,244,240,96
1110 DATA 138,138,138,72,72,8,8,8,9
0,85,85,85,68,192,192,192,165,170,1
06,0,0,0,0,0
1120 DATA 90,90,154,19,19,19,19,
32,152,152,152,152,152,168,42,8,38,
38,38,38,38,38,168
1130 DATA 162,162,171,43,62,61,60,2
55,138,138,234,233,189,125,60,255,8
,42,234,251,127,95,15,63
1140 DATA 64,16,21,17,81,87,5,61,4,
16,80,16,20,84,64,240,0,0,0,0,0,48,
12,3
1150 DATA 63,63,63,58,58,50,48,240,
240,244,245,185,186,186,48,60,3,3,6
7,87,85,165,5,85
1160 DATA 0,0,0,0,2,9,25,38,0,0,0,1
70,106,153,157,111,0,0,0,128,96,152
,159,102
1170 DATA 38,41,41,10,2,0,0,3,110,1
57,174,110,191,191,192,192,102,154,
154,104,160,128,192,240
1180 DATA 0,0,0,255,40,40,248,56,0,
0,0,0,0,5,21,84,9,42,43,10,0,0,0,0
1190 DATA 90,248,255,168,12,60,0,0,
137,42,170,170,170,42,58,58,90,106,
169,165,169,170,3,3
1200 DATA 138,168,162,106,170,170,1
60,160,0,0,0,160,160,128,0,0
1210 DATA 0,0,1,6,26,27,107,107,0,8
5,170,170,170,235,235,235,0,0,64,14
4,164,228,233,233
1220 DATA 106,106,27,26,6,1,0,0,170
,170,170,235,190,170,85,0,169,169,2
28,164,144,64,0,0
1230 DATA 255,255,63,60,15,3,0,0,25
5,195,60,255,255,255,255,0,255,255,
252,60,240,192,0,0
1240 DATA 60,60,60,60,60,60,60,60
1250 DATA 12,56,40,40,235,235,235,4
0,40,235,235,235,40,40,40,40,0,0,0,
0,255,255,255,0

```

```

1260 DATA 0,255,255,255,0,0,0,0,60,
60,60,60,60,60,149,149,149,170,60,6
0,60,60,60,60
1270 DATA 0,0,3,15,63,63,252,252,0,
255,255,255,255,60,60,60,0,0,192,24
0,252,252,63,63
1280 REM * CHANGE DISPLAY LIST *
1290 DL=PEEK(560)+PEEK(561)*256:POK
E DL+3,68:FOR J=6 TO 24:POKE DL+J,4
:NEXT J:POKE DL+25,7:POKE DL+26,6
1300 POKE DL+27,6:FOR J=29 TO 32:PO
KE DL+J-1,PEEK(DL+J)
1310 RESTORE 1320:FOR J=708 TO 712:
READ A:POKE J,A:NEXT J
1320 DATA 72,62,52,78,58
1330 POKE 752,1:RETURN
1340 REM * SCRAMBLE ANIMAL POS. *
1350 FOR J=0 TO 8
1360 R=INT(RND(0)*9):FOR K=0 TO J-1
:IF A1(K)=R THEN 1360
1370 NEXT K:A1(J)=R:A2(J)=A1(J)+ANI
MAL:IF A2(J)>8 THEN A2(J)=A2(J)-9
1380 NEXT J:RETURN
1390 REM * PRINT ANIMAL ON RIGHT *
1400 RESTORE 1440+10*ANIMAL2:READ A
$:POSITION X2(A2(J)),Y2(A2(J)):? A$
;
1410 READ A$:POSITION X2(A2(J)),Y2(
A2(J))+1:? A$:RETURN
1420 REM * PRINT ANIMAL ON LEFT *
1430 RESTORE 1440+10*ANIMAL:READ A$
:POSITION X1(A1(J)),Y1(A1(J)):? A$;
1440 READ A$:POSITION X1(A1(J)),Y1(
A1(J))+1:? A$:RETURN
1450 DATA , ,<=, ,
1460 DATA >?AB,CDEF ,
1470 DATA GH,IJK,
1480 DATA LMN,OPQ,
1490 DATA RST,UUV,
1500 DATA XY,ZI,
1510 DATA ,\J_,
1520 POKE 764,255:POKE 18,0:POKE 19
,0:POKE 20,0:RETURN
1530 TIME=PEEK(20)+256*PEEK(19):IF
TIME>MAXTIME THEN MAXTIME=TIME
1540 RETURN
1550 P=2*N:FOR J=15 TO 0 STEP -0.6:
POSITION P,20:? #6;CHR$(N+144):POSIT
ION P,20
1560 ? #6;" ":SOUND 3,40,10,J:NEXT
J:POSITION P,20:? #6;N:RETURN
1570 FOR J=5 TO 17:POSITION 2,J:? B
$:NEXT J:RETURN

```



```

1580 FOR J=5 TO 17:POSITION 21,J: ? #6;" COUNT AN
B$;:NEXT J:RETURN
1590 TIMEUP=0:POKE 764,255
1600 TIME=PEEK(20)+256*PEEK(19):IF
TIME>MAXTIME THEN TIMEUP=1:RETURN
1610 IF PEEK(53279)=3 THEN POP :GOS
UB 1830:GRAPHICS 0:GOSUB 1290:POKE
756,CSPAGE:GOTO 190
1620 IF PEEK(764)=255 AND STRIG(0)=
1 THEN 1600
1630 RETURN
1640 REM * FROWN *
1650 POKE 1667,1
1660 P=4*(8-FROWN)+2:POSITION P,1: ? #6;"qrs":POSITION P,2: ? #6;"ghi":F
ROWN=FROWN+1:IF FROWN=9 THEN 190
1670 IF SMILY+FROWN>9 THEN SMILY=SM
ILY-1
1680 RETURN
1690 REM * SCRAMBLE ARRAY *
1700 FOR J=0 TO LIM-2
1710 GOSUB 610:FOR K=0 TO J-1:IF B1
(K)=R THEN 1710
1720 NEXT K:B1(J)=R:NEXT J:RETURN
1730 FOR J=0 TO LIM-1
1740 GOSUB 630:FOR K=0 TO J-1:IF B1
(K)=R THEN 1740
1750 NEXT K:B1(J)=R:NEXT J:RETURN
1760 REM * SMILY *
1770 POKE 1666,1
1780 P=4*SMILY+2:POSITION P,1: ? #6;"abc":POSITION P,2: ? #6;"def":SMILY
=SMILY+1
1790 IF SMILY=9 THEN FOR J=1 TO 400
:NEXT J:POKE 1664,1:POKE 1665,1:GOT
O 190
1800 IF SMILY+FROWN>9 THEN FROWN=FR
OWN-1
1810 RETURN
1820 REM * MENU *
1830 GRAPHICS 17:POSITION 3,3: ? #6;"SETUP CONDITIONS":POSITION 3,5: ? #6;"choose #"
1840 POSITION 1,8: ? #6;"(1) locatio
ns":POSITION 4,9:IF LOCN=0 THEN ? #6;" ORDERED"
1850 IF LOCN=1 THEN ? #6;" RANDOM"
1860 POSITION 1,11: ? #6;"(2) animal
s":POSITION 4,12:IF TYPE=0 THEN ? #6;" SAME"
1870 IF TYPE=1 THEN ? #6;" DIFFEREN
T"
1880 POSITION 1,14: ? #6;"(3) task":
POSITION 4,15:IF TASK=0 THEN ? #6;"
MATCH GROUPS"
1890 IF TASK=1 THEN ? #6;" COUNT AN
IMALS"
1900 IF TASK=2 THEN ? #6;" MATCH PA
TTERN"
1910 POSITION 1,17: ? #6;"(4) dly ti
me ";DLY:POSITION 1,19: ? #6;"(5) #
limit ";LIM
1920 POSITION 1,21: ? #6;"(6) start
pgrm"
1930 GET #1,A:IF A<49 OR A>54 THEN
1920
1940 IF A=54 THEN RETURN
1950 GOSUB 1950+10*(A-48):GOTO 1830
1960 LOCN=ABS(LOCN-1):RETURN
1970 TYPE=ABS(TYPE-1):RETURN
1980 ? #6;CHR$(125):POSITION 3,6: ? #6;"choose a task":POSITION 1,8: ? #6;"(1) MATCH GROUPS"
1982 POSITION 1,10: ? #6;"(2) COUNT
ANIMALS":POSITION 1,12
1984 ? #6;"(3) MATCH PATTERN":GET #
1,A:IF A<49 OR A>51 THEN 1980
1986 TASK=A-49:RETURN
1990 ? #6;CHR$(125):POSITION 3,6: ? #6;"ENTER DELAY":POSITION 3,8: ? #6;"TIME (SEC)":POSITION 3,10: ? #6;"(1
-9)";
1992 GET #1,DLY:DLY=DLY-48:IF DLY<1
OR DLY>9 THEN 2020
1994 RETURN
2000 ? #6;CHR$(125):POSITION 3,6: ? #6;"ENTER MAX.":POSITION 3,8: ? #6;"NUMBER OF":POSITION 3,10: ? #6;"ANIM
ALS (3-9)"
2010 GET #1,LIM:LIM=LIM-48:IF LIM<2
OR LIM>9 THEN 2050
2020 RETURN
2030 GET #1,DLY:DLY=DLY-48:IF DLY<1
OR DLY>9 THEN 2020
2040 RETURN
2050 ? #6;CHR$(125):POSITION 3,6: ? #6;"ENTER MAX.":POSITION 3,8: ? #6;"NUMBER OF":POSITION 3,10: ? #6;"ANIM
ALS (3-9)"
2060 GET #1,LIM:LIM=LIM-48:IF LIM<3
OR LIM>9 THEN 2050
2070 RETURN
2080 GRAPHICS 18:RESTORE 2090:FOR J
=1 TO 12:READ X,Y,A:POSITION X,Y: ? #6;CHR$(A):NEXT J:RETURN
2090 DATA 7,3,111,9,2,76,11,3,196,1
,8,205,3,7,65,5,8,99,8,8,228,10,7,2
39,12,8,110,14,7,193,16,8,236,18,9,
68

```

GTIA Sketchpad con't

```

32703 DATA 219,78,221,70,217,70,248
,85
32704 DATA 38,153,36,153,36,145,17,
72
32705 DATA 4,65,4,32,0,16,0,0
32708 FOR X=79 TO 0 STEP -1:P=5:FOR
Y=0 TO 191:LOCATE X,Y,A:PTR=A+1:
BUFFER$(P)=GREY$(PTR,PTR+1):P=P+2:N
EXT Y
32709 PRINT #7;" ";BUFFER$
;"

```

Epson MX80 printer routine

```

32699 REM GR. 9 DUMP PROGRAM FOR MX
80 BY RON MILLER, SAN DIEGO ACE
32700 TRAP 32707:DIM GREY$(32),BUFF
ER$(400):OPEN #7,8,0,"P:"
32701 FOR I=1 TO 32:READ A:GREY$(I,
I)=CHR$(A):NEXT I:REM Set grey scal
e
32702 DATA 255,255,251,223,251,222,
206,183
32703 DATA 219,78,221,70,217,70,248
,85
32704 DATA 38,153,36,153,36,145,17,
72
32705 DATA 4,65,4,32,0,16,0,0
32706 BUFFER$(1)=CHR$(27):BUFFER$(2
)="K":BUFFER$(3)=CHR$(128):BUFFER$(
4)=CHR$(1):REM Tell Epson 384 dots
coming
32707 PRINT #7:PRINT #7:PRINT #7;CH
R$(27);"A";CHR$(8):REM set Epson fo
r 8 dots/line
32708 FOR X=79 TO 0 STEP -1:P=5:FOR
Y=0 TO 191:LOCATE X,Y,A:PTR=A+1:
BUFFER$(P)=GREY$(PTR,PTR+1):P=P+2:N
EXT Y
32709 PRINT #7;" ";BUFFER$
:NEXT X
32710 RETURN

```

PLEASE send us the screen dump routines for your printer that will work with this program or other programs.

REMAKER/REMOVER

REMAKER is written to help me with some of the download files I've gotten which are documentary types. Not all of them are really transportable to an ATARI system, because they don't have line numbers. REMAKER adds a line number and REM statement in front of each line of the candidate file. There are a lot of really useful things resulting from this process.

First, if you don't have an Assembler/Editor you can display or edit an Assembly language program without the BASIC interpreter going wild and rejecting the program. Second, this program has handled a number of files which none of the text editors I own will load or handle successfully. Third, since Kurt Holter told me he was investigating the translation of some Apple software on disk, I wrote the program to handle 256-character lines which is the Apple logical line maximum length.

REMOVER is a program to remove the line numbers and REM statements added by REMAKER. This program combination permits you to enter a program into BASIC, use the BASIC editor, which is very powerful and easy to use (told you it was the best language), make your changes or additions and deletions, and then recreate the original file in an edited form. To the best of my knowledge neither program has serious bugs of any kind.

—Bob Schniebold

REMAKER

& REMOVER

```
10 REM * A FREDERICK ATARI COMPUTER
* * TECHNICAL SOCIETY ORIGI
NAL * * PROGRAM
*
```

```
20 REM REMAKER by Robert Schniebold
REMAKER processes files w
hich are not BASIC compatib
le by add
```

```
30 REM ing line numbers and REM sta
te- ments. The output file ca
n then be edited by the scree
n editor.
```

```
40 REM The output file's name will
be the same as the source, b
ut it will have an extension
'REM'.
```

```
50 REM NOTE: REMAKER WILL NOT PROCE
SS 'SAVED' FILES PROPERLY. I
T IS INTENDED FOR EASY EDIT
ING OF
```

```
60 REM DOWNLOADED FILES OF ALL TYPE
5. 01/09/83.....
```

```
..RJ5
70 POKE 82,0:GRAPHICS 0:?, "REMAKER
-by Bob Schniebold"
```

```
80 DIM A$(256),B$(2),I$(15),O$(15):
B$=" ":A=98:L=LEN(A$)
```

```
90 CLOSE #1:CLOSE #2:POSITION 0,4:
"ENTER INPUT FILESPEC ":INPUT I$
```

```
100 FOR K=1 TO LEN(I$):IF I$(K,K)="
" THEN B=K
```

```
110 NEXT K:IF B THEN O$=I$(1,B):O$(
B+1)="REM":GOTO 130
```

```
120 O$=I$:O$(LEN(O$)+1)=" REM"
130 TRAP 90:OPEN #1,4,0,I$:OPEN #2,
8,0,O$
```

```
140 ? :? "PROCESSING FILE":B=115:TR
AP 190
```

```
150 INPUT #1,A$:A=A+2:IF L<=B THEN
? #2;A;B$;A$
```

```
160 IF L>B AND L<=2*B THEN ? #2;A;B
$;A$(1,B):A=A+2: ? #2;A;B$;A$(B+1)
```

```
170 IF L>2*B THEN ? #2;A;B$;A$(1,B)
:A=A+2: ? #2;A;B$;A$(B+1,2*B):A=A+2:
```

```
? #2;A;B$;A$(2*B+1)
180 GOTO 150
```

```
190 CLOSE #1:CLOSE #2: ? CHR$(253):?
"TASK COMPLETED": ? :TRAP 210
```

```
200 ? " - HIT RETURN TO": ? "EN
TER ":CHR$(34);O$;CHR$(34):POSITION
0,9:NEW
```

```
10 REM * A FREDERICK ATARI COMPUTER
* * TECHNICAL SOCIETY ORIGI
NAL * * PROGRAM
*
```

```
20 REM REMOVER by Robert Schniebold
REMOVER processes files w
hich have been created by R
EMAKER to
```

```
30 REM remove the line numbers and
REM statements produced by RE
MAKER. It's output returns th
e file to
```

```
40 REM its original form after edit
ing is completed. REMOVER's o
utput file name is the same
as the
```

```
50 REM source file, but does not ha
ve an extension.....RJ5 01/
09/83
```

```
60 POKE 82,0:GRAPHICS 0:?, "REMOVER
-by Bob Schniebold": ? : ? : ? : ?
```

```
70 DIM A$(120),I$(15),O$(11)
80 CLOSE #1:CLOSE #2: ? "ENTER INPUT
FILESPEC ":INPUT I$:
```

```
90 IF I$(LEN(I$)-2)<>"REM" THEN ? C
HR$(253);"FILE MUST HAVE 'REM' EXTE
NTION PRODUCED BY REMAKER!": ? :GOTO
```

```
80
100 O$=I$(1,LEN(I$)-4)
```

```
110 TRAP 80:OPEN #1,4,0,I$:OPEN #2,
8,0,O$
```

```
120 ? :? "PROCESSING FILE": ? :TRAP
150
```

```
130 INPUT #1,A$:IF LEN(A$)>5 THEN ?
#2;A$(6): ? A$(6)
```

```
140 GOTO 130
```

```
150 CLOSE #1:CLOSE #2: ? CHR$(253):?
"TASK COMPLETED":TRAP 200
```

RUTH'S PILOT con't

```
1030 50:13
```

```
1040 PA:10
```

```
1050 C:@B764=49
```

```
1060 50:0
```

```
1070 J:*BEGIN
```

```
1080 E:
```

```
2000 *SOUND
```

```
2010 50:13
```

```
2020 PA:10
```

```
2025 C:@B764=49
```

```
2030 50:0
```

```
2040 J:*BEGIN
```

```
2050 E:
```


Ocker's Assembly Language Listings

PAGE 2

UBI SOUND ASSEMBLY LISTING BY Stan Ockers

PAGE 1

```

10 ; SOUND ROUTINE FOR UBI
=D200 20 AUDF1 = $D200 ; AUDIO FREQ. REGISTER
=D201 30 AUDC1 = $D201 ; DISTORTION-VOL REGISTER
0000 40 *= $0680
0680 50 REPS *= *+4 ; (1664) * TIMES THRU
0684 60 CNT5 *= *+4 ; (1668) NEXT NOTE DELAY
0688 70 STRLO *= *+4 ; (1672) SOU. STRING LO
068C 80 STRHI *= *+4 ; (1676) SOU. STRING HI
0690 90 STRPOS *= *+4 ; (1680) POS. IN STRING
0694 0100 DIST *= *+4 ; (1684) DIST-VOL BYTES
0698 0110 SUSH *= *+4 ; (1688) PRESENT LOUDNESS
069C 0120 LOUD *= *+4 ; (1692) MAX. LOUDNESS
06A0 0130 QUIET *= *+1 ; (1696) CNT TO START QUIET
=00CB 0140 INDLO = $CB ; (203) INDIRECT POINTER
=00CC 0150 INDHI = $CC ; (204)
=00CD 0160 SAVEX = $CD ; (205) TEMP STORAGE
=00CE 0170 SAVED = $CE ; (206) " "
=E462 0180 XITVB = $E462 ; BACK TO ATARI UBI
06A1 0190 *= $00
0000 A203 0200 LDX #003 ; 4 voices
0002 B08006 0210 LOOP LDA REPS,X ; is voice active?
0005 F073 0220 BEQ NEXT ; no
0007 DEB406 0230 DEC CNT5,X ; load new note?
000A 3027 0240 BMI NOTE ; yes
000C B08406 0250 LDA CNT5,X ; reduce volume?
000F CDA006 0260 CMP QUIET
0012 B066 0270 BCS NEXT ; no
0014 BD9806 0280 LDA SUSH,X ; finished quiet?
0017 F061 0290 BEQ NEXT ; yes
0019 BD9406 0300 LDA DIST,X ; take dist-vol
001C 29F0 0310 AND #F0 ; mask out vol
001E 1D9806 0320 ORA SUSH,X ; put new vol in
0021 48 0330 PHA ; save for later
0022 B6CD 0340 STX SAVEX ; keep index
0024 BA 0350 TXA ; double index
0025 0A 0360 ASL A
0026 AA 0370 TAX
0027 68 0380 PLA ; get byte back
0028 9D01D2 0390 STA AUDC1,X ; reload dist-vol
002B A6CD 0400 LDX SAVEX ; recover index
002D DE9806 0410 DEC SUSH,X ; next vol level
0030 18 0420 CLC ; skip over
0031 9047 0430 BCC NEXT
0033 B08806 0440 NOTE LDA STRLO,X ; string address
0036 B5CB 0450 STA INDLO ; into indirect
0038 B08C06 0460 LDA STRHI,X ; pointer
003B B5CC 0470 STA INDHI
003D BD9406 0480 LDA DIST,X ; save dist-vol
0040 B5CE 0490 STA SAVED ; for later
0042 BC9006 0500 LDY STRPOS,X ; index into string
0045 B6CD 0510 STX SAVEX ; save index
0047 BA 0520 TXA ; double index
0048 0A 0530 ASL A
0049 AA 0540 TAX
004A B1CB 0550 LDA (INDLO),Y ; get byte
004C F020 0560 BEQ REST ; rest if zero
004E 9D00D2 0570 STA AUDF1,X ; use as freq
0051 ASCE 0580 LDA SAVED ; get dist-vol
0053 9D01D2 0590 REST1 STA AUDC1,X ; update register
0056 A6CD 0600 LDX SAVEX ; recover index
0058 C8 0610 INY ; next byte

```

```

0059 B1CB 0620 LDA (INDLO),Y ; is duration
005B F017 0630 BEQ FIN ; if zero, finished
005D 9D8406 0640 STA CNT5,X ; else, update counter
0060 C8 0650 INY ; index for next byte
0061 98 0660 TYA ; and save index
0062 9D9006 0670 STA STRPOS,X
0065 BD9C06 0680 LDA LOUD,X ; reset loudness
0068 9D9806 0690 STA SUSH,X ; counter
006B 18 0700 CLC ; skip over
006C 900C 0710 BCC NEXT
006E 9D00D2 0720 REST STA AUDF1,X ; sound off
0071 18 0730 CLC
0072 90DF 0740 BCC REST1
0074 9D9006 0750 FIN STA STRPOS,X ; reset string pos.
0077 DEB006 0760 DEC REPS,X ; next repetition
007A CA 0770 NEXT DEX ; next voice?
007B 1085 0780 BPL LOOP ; yes
007D 4C62E4 0790 JMP XITVB ; return to Atari UBI

```

PAGE 3

SYMBOLS

=D201 AUDC1	=D200 AUDF1	0684 CNT5
=00CC INDHI	=00CB INDLO	0002 LOOP
0033 NOTE	06A0 QUIET	0680 REPS
=00CE SAVED	=00CD SAVEX	068C STRHI
069B SUSH	=E462 XITVB	
0694 DIST	0074 FIN	
069C LOUD	007A NEXT	
006E REST	0053 REST1	
0688 STRLO	0690 STRPOS	

TWO BASIC COMPILERS

by Pat Warnshuis, Editor, Portland Atari Club
(published in ACE and PAC simultaneously)

Two compilers for Atari BASIC have appeared on the market almost simultaneously:

The BASIC Compiler, by DATASOFT, INC. 9421 Winnetka Avenue, Chatsworth, CA 91311. 48K, disk.

ABC: A BASIC Compiler, by Monarch Data Systems, PO Box 207, Cochituate, MA 01778. 40K, disk. \$69.95.

A BASIC compiler converts your SAVED BASIC program to fixed machine language code which is run as a binary program. The BASIC cartridge is removed as there is no need to re-interpret your code every time it is executed. Normally, when your BASIC program is RUN, the BASIC interpreter picks up each instruction from your program in sequence and asks "What do you want me to do now?" Then it goes off to perform the task you assigned to it. It also has to look up where it saved any variables, numbers, or strings as they occur in your code. Lots of error-checking going on also to keep the program from running amuck. The RUN-time interpreter has to figure out all these things every time it executes the same line of your code.

The great advantage of this method is the process is entirely interactive. That is, you tell BASIC to do something and it does it immediately. You see the results of your instructions right now and can instantly stop BASIC to change your program. Atari BASIC is the most generous BASIC of all in the freedom it allows you to stop, change, and continue your program as it is running. It is also one of the best for telling you immediately if you made a mistake while typing in your code. But the price you pay for this ease of interactive programming and error checking is time.

A compiler runs much faster because it does not have to interpret each line of code every time it executes it. It figures out what you want to do just once. Also, it does all the bookkeeping and table lookup just once. So when you run your program it already knows what you want to do and where everything is. It simply says "Do it! Do it! Do it! Don't look around! Don't get fancy! ... Just do it! Do it! Do it!" It doesn't have to ask itself all those complicated questions about "what are you talking about and where did I put that thing?" A compiled program can run much faster than your BASIC interpreter because it is screaming through your program in a direct line in machine language with everything figured out in advance.

But the price you pay for time is convenience. You don't see your errors until the compiler tries to translate your source statements into its machine language (compile-time errors). First you enter all of your code at one time using a program editor which will accept anything you give it. Then you compile the entire program in one operation. That's the first time you trap all those silly syntax and typing errors. The compiler will report most of them as it tries to make sense of your code. Many times, however, a single error confuses the compiler so much it simply has to give up right then and there. Then you go back to your program editor, correct the mistake, and try to compile the complete program again. Things are happening in big chunks here and this can be frustrating for a hacker who has been spoiled by an interactive interpreter such as BASIC.

Once your program is compiled, it sits in a disk file all by itself. It's just another machine language program. You load it and run it as a binary program without the BASIC cartridge. The compiler does slip in a lot of its own routines to help execute your statements at run-time. But these are invisible to you.

When you run your compiled program, you will discover your run-time errors. These are the errors which slipped through at compile time because their syntax was OK and the typing looked fine, but the program suffers from a bad case of aphasia. "My computer keeps doing what I tell it to do, instead of what I want it to do!" Now we see another big difference in running our program under the BASIC interpreter as opposed to compiled code. Remember, the BASIC interpreter is always there. It lets us stop, change, and continue at statement in our program if we want to. When we discover a run-time error in our compiled program, however, we have to stop, reload the editor, reload our source code, make the correction, recompile the program, save it, and try to run it once more.

Of course, we soon get smart in using a BASIC compiler. We learn to write, debug, and run the program entirely with the BASIC interpreter before we try to compile it for the first time. Some operations will be so time critical we can't do that: animated or complex graphics, real-time simulations, fast sampling measurements, or process controllers, for example.

So, while a compiled program will run some three to fifteen times faster than an interpreted program, it is more difficult to write, debug and change. Generally, I say, "Don't use a compiler until you've proven the interpreter can't do the job." But, when you must have the increased speed or memory, a BASIC compiler offers a solution which may be simpler than other alternatives, such as Forth, Pascal, C, or assembly language programming.

These two compilers from DATASOFT and Monarch Data Systems differ sufficiently in their methods and capabilities and applications. You will have to evaluate them in terms of your peculiar requirements. The following two reviews are intended to help you make your choice.

DATASOFT BASIC COMPILER

DATASOFT's compiler generates an intermediate assembly language source listing which is assembled into 6502 machine code with a 3-pass assembler. When your resultant binary program is saved to disk the compiler merges it with a run-time package of standard routines needed to execute your code.

Don't be surprised if you compile a single line program and see your binary file takes some 34 sectors! DATASOFT merges its entire run-time package with each program you compile, regardless of the size of your program. As your program gets larger, the overhead becomes proportionately smaller.

A nice feature of the DATASOFT approach is it permits you to select either a floating point or a fixed point run-time package. The floating point package includes all the transcendental trig, log and exponential functions. A floating point program will run three to ten times faster than its BASIC version, depending on the amount of arithmetic in the program. The fixed point package is more limited, of course, but is extremely fast as it uses only 16-bit arithmetic over the range of -32768 to +32767. A program compiled in fixed point can run anywhere from 5 to 20 times faster than the original BASIC program. However, you lose the transcendental and random functions. Unfortunately, the compiler does not offer you the option of mixing fixed and floating point variables. (But then, it doesn't cost \$500 either.)

DATASOFT offers a striking option to proficient assembly language programmers. You can ask for a Line Reference Map which will list each line number of your BASIC program and tell you where in memory the line is assembled. There is an aphorism in the software community which states that 80% of your time is spent in 10% of your code. (Pick your own numbers.) This means you're really only hurting for time in a very small portion of your program, say in highly iterative loops or number crunching or real-time measurements and process control. So a good programmer in time trouble need only concentrate on a small portion of the assembled code. This Line Reference Map lets you zero in on the problem area for optimization.

The Line Reference Map also lets you enter the assembled program at specific line numbers, just like doing a GOTO xxxx from the direct mode in BASIC.

You can enter the assembly source code into your editor and modify it. However, I am not sure how to skip to Pass 2 of the compiler to assemble the modified version.

There are some restrictions in using the compiler. First of all, you must realize the BASIC interpreter is not resident so, naturally, you cannot use commands such as LIST, ENTER, RUN, LOAD, CONT, DOS, NEW, SAVE. They don't list them explicitly but I'm sure PRINT and LPRINT are proscribed also. Better check DEG. This could be a sneaky one to debug if not trapped. These are not limitations of the compiler, they are just the nature of the beast. One surprise, though, is you cannot use "RUN filespec" in your program, even if the called program has been compiled separately. And, since a fixed point number cannot exceed 32767, what do you do with addresses which exceed that value? Yes, it can be done and I'll leave its method to your pleasurable pursuit. It's the kind of thing which evokes a private smirk from Arlan Levitan.

You must heed some format restrictions in your BASIC program. These are not hard to live with but they do suggest you can't just load-compile-and-go with complete abandon. Watch out for nitty gritty in assigning substrings to a string variable, avoid named labels for GOSUB and GOTO's, place DATA statements at the end of your listings. Arrays cannot be dimensioned dynamically. This means you cannot say "INPUT N: DIM ARRAY(N)".

The documentation is good. The manual is clearly written so the uninitiated can walk through the compile-and-run process very nicely. Sufficient detail is provided the advanced programmer to dive in and fine tune the compiled code.

Error handling is smooth, both at compile time and at run time. The compiler flags your errors but tries to continue the assembly to the end so it can inform you of as many errors at a time as is possible.

There are a couple of bugs, one of which DATASOFT has already corrected. An ERROR 133 will be generated if you use a constant for an index in a string. You can't use "A\$(3)". You have to use "X=3: A\$(X)". If you run into this problem, send your disk back to DATASOFT and they will replace it with the corrected version.

This compiler will not boot up in a system with 64k of RAM. It will not run with the Newall Fastchip if you used the floating point option. This problem sounds suspiciously like someone entered the floating point routines at unauthorized vectors. I don't know what DATASOFT is doing about either of these two errors.

Comment: A fine implementation. Highly recommended for anyone hurting for time or space in their BASIC programs.

Ultimate Recommendation: I obtained a pirated copy for this review. (DATASOFT does not provide review copies.) I had such respect for the product I went out and purchased it, saving the pirated copy for true backup.

ABC, A BASIC COMPILER Monarch Data Systems

Monarch's compiler generates P-code rather than direct 6502 machine code. P-code is an interesting concept worthy of a future article in itself. Suffice it to say it is one level removed from machine code in that it must be interpreted somewhat like a threaded language. However, it runs very fast and is just a hint slower than machine code itself.

ABC offers fixed point arithmetic only — no floating point. So its features are quite similar to DATASOFT's compiler in the fixed point mode. However, Monarch's fixed point format is three bytes long so it can handle numbers in the range of plus or minus eight million. This is more than enough for games, graphics, file management, mail lists, modem programs, system software, etc. However, you do have to be careful in financial programs. Although you only need two decimal places for input/output, you should do all your internal calculations to at least three places to allow for roundoff. This means your financial range is more like plus or minus \$8000. Hmmm...

Interesting, too, is the point that ALL calculations are performed as integer arithmetic. What do you get in Atari BASIC for "X = 5/3*2"? Uh huh. And how about "X = INT(5/3*2)"? Now what do you expect from a fixed point system for "X = 5/3*2"? So you want to be careful about integer divide. A change to 5*2/3 is probably what you're after. But this kind of thing can be nasty to debug. Very dangerous, indeed, is the possibility in a complex calculation you don't realize you have a problem and accept bad answers unwittingly. I don't fault Monarch for this anomaly. I rather make the point that you, as a programmer, must accept the responsibility for understanding the tools of your trade.

Monarch's manual offers suggestions for scaling techniques to produce results similar to floating point. It also offers hints for simulating trig functions with fixed point using scaled lookup tables.

Surprise! Monarch's compiler includes a utility program for making your program relocatable! It involves a double compilation technique described in the literature and permits you to locate programs in high memory, or even in bank switched memory if you have a 64k board. Even without the external utility program you have a choice of where you want the load program's starting address to be.

COMPARISONS:

Unlike DATASOFT's compiler, Monarch's system allows the use of variable names with GOSUB and GOTO as well as dynamically dimensioned arrays. It also treats strings and substrings identically to Atari BASIC. DATA statements can be anywhere in your source program. Since Monarch's system is a one-pass compiler, there is far less disk swapping involved if you have a single disk system. In fact, once you load the compiler and insert your source/destination disk, you don't have to do any disk swapping.

The Monarch compiler can handle source programs up to 2000 lines long in a 48k system, or some 1000 lines in a 40k system. DATASOFT, in a 48k system, apparently has problems with programs which are larger than 100 sectors. Since you cannot chain programs, this seems like a serious limitation.

It seems there might be less effort involved in converting an existing Atari BASIC program to compiled format using the Monarch system. On the other hand, DATASOFT offers a full floating point mode with all transcendental functions. DATASOFT's integer mode in 6502 machine code should execute faster than Monarch's P-code but not a whole lot faster.

Both of these compilers are amazingly low-priced (compared to the \$350 to \$500 price tag on compilers for other systems). Neither company requires a royalty on software developed with their compilers provided you acknowledge their use in your program or documentation.

DISKWIZ

If you've been looking for a reasonably priced disk utility DISKWIZ may fit the bill. It's a menu-based program with adequate documentation allowing you to carry out a number of disk maintenance activities. It supports both Epson MX80 and NEC 8023 printers. It's also fast since most of it is written in machine language. It works with both Atari DOS 2.0 and non-DOS disks.

In addition to the "normal" disk maintenance routines such as fixing a duplicate filename and recovering a deleted file DISKWIZ performs a number of other special purpose tasks. Any disk can be mapped to produce a picture of the entire disk including bad sectors. DOS files can be traced to show sequentially which sectors make up the file. Each sector on a disk can be viewed in both hex and ASCII format. The disk directory located in sectors 361-368 can be displayed in a very readable format as well as in hex-ASCII. The data in a sector can be altered and the results written back to the disk. There is a copy routine which is much faster than DOS and not limited to DOS format disks. But because DISKWIZ requires the BASIC cartridge be installed, only 200 or so sectors can be copied at a time. Therefore 4 swaps between source and destination are required to copy an entire disk. There are some hints in the documentation on how to create "bad" sectors, but be forewarned you must take your disk drive cover off and mess with your speed control to accomplish this task.

Another set of commands allows you to tinker with the linkages of DOS sectors. Sectors can be renumbered and/or moved, the sector links can be revised after a move, the VTOC in sector 360 can be manipulated to lock out sectors from DOS. This last function can be used to fix an unformatable disk by first mapping the disk to find the unformatable sector. Then the VTOC is adjusted to fool DOS into thinking that sector is in use. The rest of the disk then becomes usable.

Finally, DISKWIZ has a series of useful options which in my mind could justify the cost of the program by themselves. There is a disk speedcheck utility for checking and adjusting the disk drive speed.

There is a hex-dec-ASCII conversion routine allowing you to rapidly convert between these three modes. There is a very good print routine allowing you to dump the contents of the screen directly with an Epson MX80 or NEC 8023 printer (including special graphics characters) or with graphics characters represented by a period with other printers. A search function can be used to find any sequence of ASCII characters anywhere on the disk. And there is an onboard disassembler which can be used to disassemble the disk sector by sector to the screen and get a printout of the disassembled code if desired. The disassembly can be offset between 0-6 bytes and the last three bytes of each sector can be ignored (this is where the link pointer and bytes in use information are stored).

The disadvantages I see in DISKWIZ are the assumptions it makes about the level of knowledge about disk structure which may be above the level of the average user and the documentation is a bit obscure at times. Neither of these problems are insurmountable though and I recommend the program if you're interested in finding out more about your Atari disks and also more about the programs on those disks.

DISKWIZ, by Jerry Allen, is available from Allen Macroware, 1906 Carnegie Lane "E", Redondo Beach, CA 90278.

— Tony Vander Heide
Corvallis, Oregon

TINY TURT

by Ruth Ellsworth

One of the nicest things about ATARI PILOT is turtle graphics. The ease of doing graphics with the turtle makes PILOT an especially attractive language for young children.

In the event you have not met him, the turtle is a friendly little fellow. One has only to point him in the direction desired and tell him how far to go. This eliminates the necessity of plotting pictures out before hand on graph paper, a chore which the younger children in our home greet with the same enthusiasm as a trip to the dentist's office.

Because the turtle is invisible, it is sometimes difficult to help children visualize where the turtle is, and where he is going. We solved the problem in our home by using the Visiurt program by David W. Thornburg published in **COMPUTE!** magazine, April 1982. If you do not have access to that magazine, you can use a method similar to LOGO by writing a simple program which allows the child to direct the turtle with words like up, down, left, right, etc. A sample of such a program is listed at the end of this article.

One of the nicest things about the turtle is he can give children a feeling or orientation to geometric and mathematical concepts. When children do not know what they need to do to make the turtle draw the picture they want, they can be encouraged to pretend to be the turtle and pace out the desired shape. It is an exciting and joyful experience to a child to discover how geometric shapes are made by pretending to be the turtle. Relationships which otherwise might remain a mystery or have to be memorized in some laborious fashion truly become "child's play." Such "discovery learning" seems to be transferred into other applications with greater ease than traditional methods of approaching similar concepts.

Since children learn best when they have someone to share the learning experience, I recommend you program with them. If you are just a beginner, you may want to try some simple line drawings using the LOGO type program. The documentation included in the PILOT package can be used to choose degrees of turning to make more complicated pictures in your own programs.

The second and longer program included in this article is one I wrote for my just turned three year old. He wanted something that allows him to actually make something he can understand happen at the computer keyboard. This little program allows him to draw pictures using the directional arrow keys and the $\times$, $\uparrow$, $\downarrow$, and $\rightarrow$ keys. The special keys make diagonal lines, which I have left as he likes ("fancy"); the $\times$ and $\uparrow$ being above the arrows draw diagonal lines going up, and the $\downarrow$ and $\rightarrow$ draw diagonal lines going down. The turtle is double size and flashes so my little one can tell where he is. The program does not allow him to go "out of bounds," and beeps when the edge of the picture area is reached to remind him to change direction.

The program is made up of a module for each direction. $J(\text{@B764} = x)$ means jump if at byte 764 the value = x. Byte 764 returns the value of the last key typed. Each key on the keyboard of the computer has a decimal value which can be found on page 50 of the technical manual (for those of letters only, you can refer to the Letter Snatch game published in my article in the February Newsletter). Therefore I am asking the computer to jump to the indicated module if a key is typed equal to one of the decimal values I have chosen to use in my program. No return key is necessary for input when byte 764 is used.

BANK STREET WRITER

I am ten and a half years old and was asked to review the Bank Street Writer, a home word processor for children. I think it's the best program I've worked with in my experience with computers. The manual is very thorough in giving directions, and very easy to understand. In addition, at the top of each mode, step by step directions are printed.

However, I feel the best way to learn about the Bank Street Writer is to use the tutorial program which is located on the back side of the disk. The tutorial program is a series of 5 lessons to teach you how to work the Writer. The lessons are foolproof. If you make a mistake, it is caught and you must repeat the instruction. The whole tutorial program takes approximately 30 minutes.

Once you have learned to use the Bank Street Writer, you will find it has many good features. One of the features I like best is when you type a word and it doesn't all fit on the line the computer automatically moves the whole word to the next line. Another good feature is all the options in the edit mode and transfer mode. You can erase up to 15 lines of text at one time. Then if you change your mind you can unerase it. You can also move parts of the text around. You can find a specific place in the text by using Find. Using Replace you can exchange one word for another.

In the transfer mode there are two ways to print. Print Draft and Print Final. The Print Draft prints the text exactly how it is on the screen. In Print Final you can control how the text will look. And the neatest thing is it asks if you really want to do the option you choose before it does them.

The only bad thing is I couldn't figure out how to indent the first line of a paragraph using the indent feature. But the space bar worked just fine. I recommend this program for all ages. Even my dad had fun with it. It is really AWESOME!

AND NOW A WORD FROM MOM:

For you parents who are novices with the computer as I am, take heart. The Bank Street Writer is so easy to understand and use that even we can use it. It is the perfect tool for gaining confidence on the computer as well as being an excellent word processor for a beginner. I spent only minimal time with Wendy while she learned how to use the Bank Street Writer. After doing the lessons on the tutorial program, I too felt very at ease with the processor. I wholeheartedly endorse this for any family with school age children.

—Wendy and Sandy Cheldelin

ERACE UPDATE

I have just finished compiling and editing a list of all the educational programs in the ACE library. I copied the best of these programs onto separate disks, and we now have three disks devoted to educational material. If you want a listing of these disks with program descriptions, send a SASE to ERACE, c/o Robert Browning, 90 W. Myoak Dr., Eugene, Ore., 97404.

We want to thank Roy Dugan for the three programs he sent us last month. These programs, Memory, Numbers 15, and Math, are included on one of the educational disks. (Note: we will have all of the programs on cassette also.)

Our May meeting will be held at Ali Erickson's, 295 Foxtail Dr., on the 3rd of May. This will be a public domain educational software swap. So bring your programs, your empty disks and tapes, and come share with us. Even if you don't have any public domain programs to swap, come and start building your library. If you can't make it to the meeting, but you still want to swap material; let us know what you have and what you're looking for and we'll see what we can do.

The June meeting will be on the 7th, at a place to be announced. We will be evaluating programs in our library, plus any new material we receive.

New users often ask us, "Now that I have my Atari what do I do with it?" So we are taking on a new project, and we are soliciting your input. We want to put together a list of materials, hardware, software, literature, etc., for people just starting with their Atari. If you have something you think you can't do without, or know of something you wish you had had when you were starting, let us know.

We are still soliciting software for review. Even though we want to see as much commercial work as possible, we are just as interested in seeing your pet project even if you don't plan to market it.

Happy learning until next month.

—Bob Browning

ERACE ERACE SOFTWARE EVALUATION FORM

The First Draft

ERACE members have been putting together a software evaluation form. It is our first product oriented project and a draft copy is included in this newsletter. I think at times we all felt we were most probably reinventing, or at least reshaping, the wheel. However, the process of doing this together got individuals from very diverse backgrounds communicating in a common language about software. The process was thought provoking, stimulating and educational. The essence of a project like this, and perhaps user groups in general, is we got together and taught ourselves something about "the state of the art". We are looking forward to testing out our evaluation form on our own software library as well as on other public domain and commercial software. Any comments and suggestions will be appreciated. THANKS TO ALL ERACE MEMBERS WHO WORKED ON THIS!

—Ali of ERACE

BANK STREET WRITER

Program Review by Robert Browning (ERACE)

"The Bank Street Writer", by Broderbund Software, is an amazingly simple to use word processor. A tutorial provided on the reverse side of the disk takes you through the basic functions of the processor and step-by-step shows you how they work. My nine year old son was able to use the processor after going through the tutorial.

The screen is formatted so it is very easy to read and understand. All functions and instructions are thoroughly prompted on the screen.

The program has excellent documentation which thoroughly explains each function of the processor including some of the finer points not in the tutorial.

The Bank Street Writer is very well designed and appears to have taken into consideration users of all ages. No matter what I have done to the program it has never crashed. This one appears to be kid-proof, or is that user-proof?

It has all the writing and formatting options needed to produce a document of moderate length. But keep in mind this is not meant to be a full-blown word processor. It does not have full printer controls, nor many of the other complex formatting options you get with the more complicated word processors. Some users feel this is a big deficit, but I think those who will use this writer probably will not need those options. It is designed so anyone, including children, can use it to easily write small documents such as letters, reports and articles.

The only real drawback is there is only room for 1000 characters with the Basic cartridge in, and 2000 characters without the cartridge. But even this can be overcome by periodically filing the text and then combining the files when the document is printed. Moving the cursor during the edit functions is somewhat awkward and takes some getting used to. But then again it's something any nine year old child can do.

I used Bank Street Writer to write this review. Even though it does have limited space, and the cursor control is a little cumbersome at first, I liked it for its simplicity and ease of use. I give it 7.5 on a scale of 1 to 10.

It takes 48K and a disk drive to use the Bank Street Writer, and sells for \$69.95.


```

0 REM REPRINTED FROM ACE OF COLUMBUS, OHIO
1 K=300
4 REM *****
5 REM * STING BY BRENT BORGHESE *
6 REM *****
7 FOR R=0 TO 33:READ Z:POKE 1538+R,Z:NEXT R
10 READ A
21 IF A=255 THEN END
25 REM SOUND 0,A,10,10
30 Z=USR(1538,255-(A*1.3),A)
50 FOR L=1 TO 27:NEXT L
60 GOTO 10
62 DATA 104,104,104,133,205,104,104,133,204
,169,255,141,31,208,164,204,136,208,253,169
63 DATA 0,141,31,208,164,204,136,208,253,19
8,205,208,232,96
70 DATA 108,102,96,60,86,60,96,60,53,50,47,
80,53,47,47,64
80 DATA 53,60,108,102,96,60,96,60,96,60,72,
81,85,72,60,47,53,60
90 DATA 53,108,102,96,60,96,60,96,60,53,50,
47,60,63,47,46,64,53,60,60
100 DATA 53,47,60,53,47,47,60,53,60,47,60,5
3,47,47,60,53,60,47,60,53,47,64,53,60
110 DATA 108,102
200 DATA 108,102,96,60,96,60,96,60,53,50,47
,60,53,47,47,64
210 DATA 53,60,108,102,96,60,96,60,96,60,72
,81,85,72,60,47,53,60
220 DATA 53,108,102,96,60,96,60,96,60,53,50
,47,60,53,47,47,64,53,60,60
225 DATA 53,47,60,53,47,47,60,53,60,47,60,5
3,47,47,60,53,60,47,60,53,47,64,53,60
230 DATA 60,96,91,85
300 DATA 108,102,96,60,96,60,96,60,53,50,47
,60,53,47,47,64
310 DATA 53,60,108,102,96,60,96,60,96,60,72
,81,85,72,60,47,53,60
320 DATA 53,108,102,96,60,96,60,96,60,53,50
,47,60,63,47,47,64,53,60,60
325 DATA 53,47,60,53,47,47,60,53,60,47,60,5
3,47,47,60,53,60,47,60,53,47,64,53,60
330 DATA 60,255

```

```

2000 REM *=$600
2010 REM PLA
2020 REM PLA
2030 REM PLA
2040 REM STA      $CD
2050 REM PLA
2060 REM PLA
2070 REM STA      $CC
2080 REM BB        LOA  $FF
2090 REM STA      $D01F

```

```

2100 REM LDY
2120 REM BNE
2130 REM LDA
2140 REM STA
2150 REM LDY
2160 REM CC
2170 REM BNE
2180 REM DEC
2190 REM BNE
2200 REM RTS

```

ATR8000 INTERFACE

Software Publishers

reviewed by Pat Warnshuis, editor Portland Atari Club Newsletter
(published simultaneously in ACE and PAC)

This is the most exciting piece of hardware since the Atari 800! I am so impressed by this interface that I purchased it with two double density, double sided disk drives and sold my 850 interface with its two 810 disk drives. I cannot imagine anyone purchasing an 850 interface with an 810 disk drive now that this product has arrived.

The ATR8000 starts out as an interface unit similar in function to the Atari 850 interface. In fact, you simply unplug the 850 interface and plug in the ATR8000 using the same cable. You get a parallel printer port, an RS-232 serial port for a modem, and an intelligent disk controller. The disk controller can address Atari 810 drives, single density single sided, double density single sided, double density double sided, either five or eight inch drives. You can mix five and eight inch drives on the same daisy chain. You can read just about any disk format: Osborne, Kaypro, Xerox, Heath, etc.

Included in the basic unit is a Z-80 4 MHz processor with 16k of RAM. The Z-80 acts as an intelligent interface so the Atari computer thinks it is talking to its own 850 interface. Part of the 16k RAM acts as a 4k print buffer. That means the interface can accept and store 4k of data as fast as the 800 can dump it out. Then it releases the 800 for other tasks while the interface's Z-80 sends the data on to the printer at the printer's comparatively slow rate.

OK. That's the basic unit. But you can add the most exotic features to it. And the more you add, the more cost effective it becomes.

First, simply change the RAM chips to get 64k of RAM. This gives you a 48k print buffer (about 20 single spaced typewritten pages).

Next get a disk with the CPM operating system on it. Now the interface thinks it is a standard CPM system with the Atari as its terminal. Run any CPM program you can find — there are thousands of them. The local RCPM bulletin board has some 100 disks of public domain, CPM User Group software available for downloading. Run Wordstar, Supercalc, dBII, PL1, Pascal, Microsoft BASIC... you can't even count 'em. Take a look at any magazine for CPM software ads.

Don't like the 40-column display of the Atari? Throw a switch and plug in any RS-232 terminal on the market. You can chose a Soroc 120, Heath 19, ADM 3A, Vision 50, Televideo... take your pick.

You say you find your Atari with double density, double sided disks, printer and modem too restrictive? Not satisfied with your 64k, 4 MHz CPM system with its 80 column, 24 line terminal? OK. Add 256k of RAM with an 8088 processor and run CPM/86 or MSDOS operating systems. That's sixteen bit processing, right up there with the Big Boys. And, get this! When you're running the Z80 with CPM, the 256k of RAM is available as a memory disk drive. Talk about speed! Hah!!!

Yeah, but how much does all this cost? Well, you can approach it in steps. Remember, the basic 16k unit with one DDDS drive gives you four times storage as an 810 drive plus a 4k print buffer just for starters. The basic unit retails for \$350; \$600 with 64K and CPM/M. Tandon Disk Drives with power supplies & cases can be obtained from independent sources for \$250.

Support? Listen, I bought Serial Number 3 of this thing. When you buy SN #3 of anything, you better have support. Software Publishers are better than great in their support. They are excited entrepreneurs who pass on their continuing discoveries of additional capabilities to us at no charge. We already have received a twelve page newsletter, several additional support software programs, and suggested mods for reduced RFI and improved RESET operation. This system is so powerful all of us will continue to discover unsuspected features and applications. Software Publishers is acting as the clearing house for a user group type newsletter.

The 8088 processor with its 128/256k of RAM already works with the Xerox 820 and the Bigboard. It will soon be available for nearly all Z80 and 8080 micros that run CPM. This translates to increased volume sales and this means more users and third party support.

The Portland Atari Club just purchased the basic 16k unit with a single DDDS drive for its bulletin board. Better get your order in. The line is going to be a long one.

DEY

```

AA
$#00
$D01F
$CC
DEY
CC
$CD
BB

```




SOFTWARE EVALUATION FORM #1
ATARI COMPUTER ENTHUSIASTS

PROGRAM DESCRIPTION

SUBJECT: _____ AGE/GRADE: _____ PROGRAM CONTENT: _____
Program Name: _____ Specific Subject Matter: _____
Producer: _____
Address: _____ Program Objectives: _____
Cost: _____
Back-up System Policy: _____ DOCUMENTATION: _____
SYSTEM: Brand _____ On Disk/Tape _____
Model _____ On Paper _____
Memory Required _____ User is taught to use program _____
Language _____ Program content is described _____
REQUIREMENTS FOR USE: _____ Program has been field tested _____
Hardware _____ Validation data is included _____
Software _____
Time _____

PREREQUISITES FOR INSTRUCTION:

Prerequisite Skills: _____
Additional Assistance Needed: _____

INSTRUCTIONAL TECHNIQUES:

_____ Tutorial
_____ Drill & Practice
_____ Educational Game
_____ Simulation
_____ Problem Solving
_____ Testing
_____ Keeps User Records

USE OF COMPUTER CAPABILITIES:

_____ Graphics _____ Color
_____ Animation
_____ Voice on Tape
_____ Computerized Speech
_____ Use of Sound Effects
_____ Variation in Timing
_____ Selection of Options
_____ Difficulty Level
_____ Sequencing
_____ Cueing and Prompting
_____ Modifiable by User

EVALUATION

[1 = Exceptional; 2 = Adequate; 3 = Unacceptable; 4 = Not Applicable]

DOES IT WORK?

#

COMMENTS:

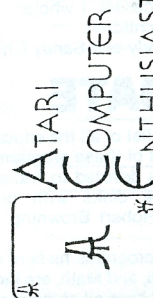
Easy to use.....
Documentation includes clear instructions....
Documentation describes content of program...
CONTENT:
Content consistent with program description..
Content based on sound instructional design..
INSTRUCTIONAL QUALITY:
Unique computer capabilities utilized.....
Presentation is polished.....
Appropriate instructional techniques
matched to stated objectives.....
Feedback is specific and motivational.....
Flexibility in presentation and timing
of content.....
COMMENTS:

MAY 11 MEETING
7.30 pm
L.C.C. FORUM 308

TYPESETTING FROM YOUR COMPUTER

ATARI OWNERS: If you have a modem, text editor, and communications program to send ASCII files, you should consider the improved readability and cost savings provided by **TYPESETTING** your program documentation, manuscript, newsletter, or other lengthy text instead of just reproducing it from line printer or daisy-wheel output. Computer typesetting by telephone offers you high quality, space-saving copy that creates the professional image you want! Hundreds of type styles to choose from with 8 styles and 12 sizes "on line." And it's easy to encode your copy with the few typesetting commands you need.

COMPLETE CONFIDENTIALITY GUARANTEED
— Bonded for your protection —
PUBLICATION DESIGN, EDITING, & PRODUCTION
Editing & Design Services
Inc.
30 East 13th Avenue Eugene, Oregon 97401
Phone 503/683-2657



3662 Vine Maple Dr Eugene OR 97405

FIRST CLASS MAIL

Atari Computer Enthusiasts

A.C.E. is an independent computer club and user's group with no connection to the Atari Company, a division of Warner Communication Company. We are a group interested in educating our members in the use of the Atari Computer and in giving the latest News, Reviews and Rumors.

All our articles, reviews and programs come from you, our members.

Our membership is world-wide; membership fees include the A.C.E. Newsletter. Dues are \$10 a year for U.S., and \$20 a year Overseas Airmail and include about 10 issues a year of the ACE Newsletter.

Subscription Dep't: 3662 Vine Maple Dr., Eugene, OR 97405
President—Kirt Stockwell, 1810 Harris #139, Eugene, Or 97403 / 503-683-3005
Vice Pres—Larry Gold, 1927 McLean Blvd., Eugene, Or 97405 / 503-686-1490
Secretary—Charles Andrews, POB 1613, Eugene, Or 974401613 / 503-747-9892
Librarian—Chuck and Jody Ross, 2222 Ironwood, Eugene, OR 97401 / 503-343-5545
Editors—Mike Dunn, 3662 Vine Maple Dr., Eugene, Or 97405 / 503-344-6193
Jim Bumpas, 4405 Dillard Rd., Eugene, Or 97405 / 503-484-9925
E.R.A.C.E. (Educational SIG Editor)—Ali Erickson, 295 Foxtail Dr., Eugene, Or 97405 / 503-687-1133

Send a business-size SASE to the Ross' for the new, updated ACE Library List!!

Bulletin Board
(503) 343-4352

On line 24 hours a day, except for servicing and updating. Consists of a Tara equipped 48K Atari 400, 2 double-density disk drives, an Atari 825 printer, a Hayes SmartModem; running the ARMUDIC Bulletin Board software written by Frank L. Huband. See the Nov '82 issue for complete details.